Deterministic and randomized local search

Citation for published version (APA):

Document status and date:
Published: 01/01/1993

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Deterministic and randomized local search

by

E.H.L. Aarts  J.H.M. Korst  P.J. Zwietering

Computing Science Note 93/04
Eindhoven, January 1993
This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology. Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review. Copies of these notes are available from the author.

Copies can be ordered from:
Mrs. F. van Neerven
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
ISSN 0926-4515

All rights reserved
editors: prof.dr.M.Rem
prof.dr.K.M.van Hee.
Deterministic and randomized local search

Emile H.L. Aarts¹,², Jan H.M. Korst¹, and Patrick J. Zwietering²

¹. Philips Research Laboratories
P.O. Box 80.000, 5600 JA Eindhoven, the Netherlands
². Eindhoven University of Technology
P.O. Box 513 5600 MB Eindhoven, the Netherlands

1. INTRODUCTION

2. LOCAL SEARCH
2.1. Formulation and modelling
2.2. Simple exchange neighbourhoods
2.3. More elaborate exchange neighbourhoods
2.4. Analysis and complexity
2.5. Related topics

3. SIMULATED ANNEALING
3.1. Threshold algorithms
3.2. Markov modelling
3.3. The homogeneous model
3.4. The inhomogeneous model
3.5. Asymptotic behaviour
3.6. Finite time approximation
3.7. Simulated annealing in practice
3.8. Related topics

4. BOLTZMANN MACHINES
4.1. Hopfield networks
4.2. Boltzmann machines
4.3. Sequential Boltzmann machines
4.4. Parallel Boltzmann machines
4.5. Combinatorial optimization and Boltzmann machines
4.6. Related topics

5. CONCLUSION

BIBLIOGRAPHY
1 Introduction

Local search is concerned with the design and analysis of a special class of algorithms that can find approximate solutions for hard combinatorial optimization problems. Solving a combinatorial optimization problem amounts to finding an optimal solution among a finite or countably infinite number of alternative solutions. Here, optimality is related to some cost criterion, which provides a quantitative measure of the quality of each solution. This area of discrete mathematics is of great practical use and has attracted much attention over the years. For extensive and detailed introductions in the field the reader is referred to the excellent books by Papadimitriou & Steiglitz [1982], Nemhauser & Wolsey [1988], and Schrijver [1986]. An annotated bibliography of the field is given by O'hEigeartaigh, Lenstra & Rinnooy Kan [1985]. Many combinatorial optimization problems belong to the class of NP-hard problems; see Garey & Johnson [1979]. It is generally believed that these problems cannot be solved to optimality within polynomially bounded computation times. This property has led to a variety of solution methods, which can be distinguished using the following criteria.

Optimization vs. approximation. Optimization algorithms find optimal solutions, at the risk of large, possibly impracticable amounts of computation time. Approximation algorithms, often called heuristic algorithms, find solutions that are not necessarily optimal within acceptable running times.

General vs. tailored. General algorithms are more or less problem independent and therefore applicable to a wide range of different problems. Tailored algorithms use problem-specific information and their use is therefore mostly limited to a restricted set of problems.

Construction vs. iteration. Constructive algorithms find solutions by constructing them by subsequent augmentation of partial solutions. Iterative algorithms find solutions by iteration over a number of complete solutions, for instance by successive improvement.

The above criteria provide a rough classification that works quite well in practice. However, there are also examples of algorithms for which it is hard to decide to which class they belong. For instance, some general algorithms are formulated as templates or "meta heuristics" with many parameters; tabu search [Glover, 1989a; 1989b] is a well-known example. Instantiation of these parameters sometimes requires knowledge that is problem specific in which case the resulting algorithm might be called tailored. For those algorithms the additional distinction between template and instantiation could resolve the seeming contradiction.

Two important performance quantifiers of algorithms are effectiveness and efficiency, which are related to solution quality and running time, respectively. Using these quantifiers, the above classification can be schematically represented as in Figure 1. At the top we have the fast optimization algorithms for the easy problems. Next in the case of hard problems, one can either opt for high effectiveness or high efficiency, which roughly speaking leads to either slow optimization algorithms or fast approximation algorithms, respectively. Finally, at the bottom, we have the slow approximation algorithms for the hard problems.

In the remainder of this chapter we discuss a class of high-quality approximation algorithms known as local search algorithms. Within the scope of the above classification, local search
algorithms can be positioned in the bottom box of Figure 1, combining the worst of both worlds, in a certain way. However, practice shows that, though slow and approximative, local search algorithms can find high-quality solutions. Local search algorithms explore neighbourhoods by comparing the cost of a current solution with that of a neighbouring solution. We distinguish between deterministic and randomized local search on the basis of the nature of the criterion that determines the acceptance of newly proposed solutions. The distinction is obvious: if the criterion is deterministic, the resulting class of algorithms is called deterministic. If the criterion is probabilistic, we speak of randomized local search.

In this chapter we review local search from a broad point of view with special emphasis on a randomized version known as simulated annealing. To this end the chapter is organized in the following three parts. First, we give a general introduction in the field of deterministic local search, addressing both algorithmic and complexity issues. Next a special class of local search algorithms called threshold algorithms is derived, which in a natural way introduces simulated annealing as a prime representative of the class of randomized local search algorithms. In the third part we discuss the role of neural networks in local search. Again a distinction is made between deterministic and randomized neural networks, which coincides with Hopfield networks and Boltzmann machines, respectively. The emphasis is on Boltzmann machines and their significance as a massively parallel approach to simulated annealing.

Figure 1: A schematic positioning of combinatorial optimization algorithms.
2 Local search

Local search algorithms constitute a class of general approximation algorithms that are based on the exploration of neighbourhoods [Papadimitriou & Steiglitz, 1982; Yannakakis, 1990]. The use of local search in combinatorial optimization already has a long history that reaches back to the late fifties and early sixties, when the first exchange algorithms were introduced for the travelling salesman problem; see the work of Bock [1958], Croes [1958], Lin [1965], and Reiter & Sherman [1965]. In the subsequent years the scope of local search has been broadened and the basic concept of exchange algorithms has been applied with some success to a variety of problems. Nicholson [1971] first discussed the application of exchange strategies to a more general class of permutation problems. The interest in the subject however was predominantly from a practical point of view and no major conceptual progress was made for some time. The past decade shows a strong renewed interest in the subject, which can be attributed to the following three developments.

Appeal. Many variations of local search based algorithms have been proposed based on analogies with processes in nature, relating the subject to other disciplines such as statistical physics, biological evolution, and neurophysiology. This has resulted in new algorithms or paradigms. Well-known examples are simulated annealing, genetic algorithms and some variations of neural networks. The analogies are sometimes taken quite far, leading to variants with extravagant names such as the great deluge algorithm by Dueck [1989], the hide and seek method by Béisle, Romeijn & Smith [1990], or the roaming ants approach by Colorni, Dorigo & Maniezzo [1992]. These names may sound extravagant, but they illustrate that the issue of problem solving by nature is attractive to many researchers.

Theory. Some of the newly proposed local search based algorithms have been mathematically modelled yielding theoretical results on their performance. Probably the best known example is the simulated annealing algorithm, that can be rigorously modelled by the theory of Markov chains. Furthermore, a complexity theory of local search has been developed which provides more theoretical insight, not only in the complexity of local search but also in part of the combinatorial structure of discrete optimization problems.

Practice. The large increase in computational resources together with the use of sophisticated data structures have made local search algorithms strong competitors within the class of algorithms designed to handle large problem instances. Furthermore the flexibility and ease of implementation of local search algorithms have been shown to be quite successful in handling complex real-world problems.

In this chapter we concentrate on the theoretical issues of local search with the emphasis on simulated annealing. Practical issues are only briefly mentioned and despite its interest the issue of appeal is discarded.
2.1 Formulation and modelling

The use of a local search algorithm presupposes the definition of a problem and a neighbourhood. These can be formulated as follows.

**Definition 2.1** A combinatorial optimization problem is specified by a set of problem instances and can be either a minimization problem or a maximization problem.

Unless explicitly stated otherwise, we consider in this chapter only minimization problems.

**Definition 2.2** An instance of a combinatorial optimization problem is a pair \((S, f)\), where the solution space \(S\) is a finite set of all possible solutions and the cost function \(f\) is a mapping \(f : S \rightarrow \mathbb{R}\).

The problem is to find a globally-optimal solution \(i^* \in S\) such that \(f(i^*) \leq f(i)\), for all \(i \in S\). Furthermore, the optimal cost is denoted by \(f^*\) and the set of optimal solutions by \(S^*\).

The careful distinction between a problem and its instantiation is made to differentiate between one single input data, the instance, from which a solution can be unambiguously obtained, and the set of input data, the problem, that have a common structure. Furthermore, the set \(S\) is generally not given explicitly, i.e., by listing all elements. Usually, one resorts to the use of a compact representation from which the elements in \(S\) can be computed by a polynomial-time algorithm [Garey & Johnson, 1979]. This representation is then chosen such that its size, e.g., the number of bits to store it in a computer, is polynomially related to the size of the input data of the instance. This size is defined as the size of the problem instance. Furthermore, the set of solutions can often be represented by a set of decision variables, whose values can have certain ranges. These variables can be directly related to the model that is used to formulate the problem. An example is given by the integer variables in integer linear programming models [Nemhauser & Wolsey, 1988; Schrijver, 1986].

To apply a local search algorithm we need the notion of a neighbourhood structure.

**Definition 2.3** Let \((S, f)\) be an instance of a combinatorial optimization problem. A neighbourhood structure is a mapping \(N : S \rightarrow 2^S\), which defines for each solution \(i \in S\) a set \(N(i) \subseteq S\) of solutions that are in some sense 'close' to \(i\). The set \(N(i)\) is called the neighbourhood of solution \(i\), and each \(j \in N(i)\) is called a neighbour of \(i\). Furthermore, we assume that \(i \in N(i)\), for all \(i \in S\).

Roughly speaking, a local search algorithm starts off with an initial solution and then continually tries to find better solutions by searching neighbourhoods.

A basic version of a local search algorithm is the one that applies iterative improvement, i.e., the neighbourhood of a current solution is searched for a solution with lower cost. If such a solution is found, the current solution is replaced by this solution. Otherwise, the algorithm continues with the current solution. It terminates when no strict improvements can be obtained any more. This algorithm is schematically outlined in Figure 2.

An important concept in the analysis of local search algorithms is that of local optimality, which can be defined as follows.
procedure ITERATIVE_IMPROVEMENT;
begin
  INITIALIZE (i_start);
  i := i_start;
  repeat
    GENERATE (j from N(i));
    if f(j) < f(i) then i := j;
  until f(j) ≥ f(i), for all j ∈ N(i);
end;

Figure 2: Pseudocode of an iterative improvement algorithm.

Definition 2.4 Let (S, f) be an instance of a combinatorial optimization problem and let N be a neighbourhood structure, then i ∈ S is called locally minimal with respect to N if

\[ f(i) ≤ f(j), \text{ for all } j ∈ N(i). \]

Furthermore, \( \hat{S} \) denotes the set of locally optimal solutions.

Definition 2.5 Let (S, f) be an instance of a combinatorial optimization problem and N a neighbourhood structure. Then N is called exact if, for each i ∈ S that is locally optimal with respect to N, i is also globally optimal.

Local search algorithms following the general scheme of Figure 2 terminate by definition in a local optimum. If the applied neighbourhood structure is exact, the local search algorithm is an optimization algorithm. It is important to note that the concept of local optimality depends on the neighbourhood structure that is used.

Alternatively, local search can be viewed as a walk in a directed, vertex-labeled graph, whose vertices are given by the elements in S and whose arcs are given by all (solution, neighbour)-pairs. The vertex labels are given by the corresponding cost values. Probably the best known example of local search in this respect is the simplex algorithm for linear programming [Dantzig, 1951; 1963], where the simplex polytope determines the neighbourhood graph and the consecutive application of pivoting rules determines the walk. Moreover, the corresponding neighbourhood structure is exact [Papadimitriou & Steiglitz, 1982; Chvátal, 1983].

2.2 Simple exchange neighbourhoods

In many combinatorial optimization problems, solutions can be represented as sequences or partitions. These solution representations enable the use of k-exchange neighbourhoods, i.e., neighbourhoods that are obtained by defining k exchanges of elements in a given sequence or
partition. Below, we discuss some examples. The definitions of the graph-theoretical concepts used in this section can be found in [Harary, 1972].

**Definition 2.6 (SORTING)** Given a multiset \( A = \{a_1, a_2, \ldots, a_n\} \) of \( n \) positive numbers. Find a non-increasing order of the numbers in \( A \).

In SORTING, \( S \) can be chosen as the the set of all permutations \( \pi \) of the numbers in \( A \) where \( \pi(i) \) denotes the number at position \( i \) in the ordering. The cost function can be chosen as

\[
f(\pi) = \sum_{j=1}^{n} j\pi(j), \quad \text{for all } \pi \in S.
\]

A \( k \)-exchange neighbourhood can be defined as follows. For all \( \pi \in S \), \( N^{(k)}(\pi) \) is given by the set of permutations in \( S \) that can be obtained from \( \pi \) by selecting \( k \) distinct pairs of numbers \((a, b), a, b \in A, \) and exchanging their positions, i.e., let \( \pi(p) = a \) and \( \pi(q) = b \), then after the exchange we have \( \pi(p) = b \) and \( \pi(q) = a \), for each of the \( k \) pairs \((a, b)\).

**Definition 2.7 (UNIFORM GRAPH PARTITIONING)** Given an undirected graph \( G = (V, E) \) with \(|V| = 2n\) and for each edge \( e \in E \) a weight \( w_e \in \mathbb{Z}^+ \). Find a partition of \( V \) into two subsets \( V_1 \) and \( V_2 \) with \(|V_1| = |V_2| = n\), such that the sum of the weights of the edges that have one end-point in \( V_1 \) and one end-point in \( V_2 \) is minimal.

In UNIFORM GRAPH PARTITIONING, \( S \) can be chosen as the the set of all partitions \((V_1, V_2)\) of \( V \) into two subsets of equal size. The cost function can be chosen as

\[
f(V_1, V_2) = \sum_{\{u, v\} \in E, u \in V_1, v \in V_2} w_{(u, v)}, \quad \text{for all } (V_1, V_2) \in S.
\]

A \( k \)-exchange neighbourhood can be defined as follows. For all \((V_1, V_2) \in S \), \( N^{(k)}(V_1, V_2) \) is given by the set of partitions in \( S \) that can be obtained from \((V_1, V_2)\) by selecting \( k \) distinct pairs of vertices \((u, v), u \in V_1, v \in V_2 \) and interchanging them, i.e., moving \( u \) to \( V_2 \) and \( v \) to \( V_1 \).

**Definition 2.8 (TRAVELLING SALESMAN)** Given a set of \( n \) cities and a distance matrix \( [D]_{n \times n} \) with elements \( d_{ij} \) denoting the distance from city \( i \) to city \( j \), \( i, j = 1, \ldots, n \). Find the shortest tour visiting each city exactly once.

In TRAVELLING SALESMAN, \( S \) can be chosen as the set of all Hamilton cycles \( C \) in the complete, weighted graph \( K_n \), whose vertices correspond to the cities and edge labels \( w_{(i,j)} \) are given by the distances \( d_{ij} \). The cost function can be chosen as

\[
f(C) = \sum_{\{i,j\} \in C} w_{(i,j)}, \quad \text{for all } C \in S.
\]

A \( k \)-exchange neighbourhood can be defined as follows. For all \( C \in S \), \( N^{(k)}(C) \) is given by the set of Hamilton cycles in \( S \) that can be obtained by removing \( k \) edges from \( C \) and replacing them with \( k \) other edges from \( K_n \).

Note that the sizes of the \( k \)-exchange neighbourhoods given in the examples above are \( \mathcal{O}(n^k) \) since in general there are \( \binom{n}{k} \) possible ways of selecting \( k \) items from a set of cardinality \( n \).
In the case of TRAVELLING SALESMAN there is an additional exponential factor of $\prod_{j=1}^{k-1} 2(k-j)$ originating from the number of different ways of replacing $k$ edges in the Hamilton cycle. The exchange neighbourhood for sorting is exact, even in the case where the exchanges are restricted to strictly adjacent numbers in an ordering. The neighbourhoods for UNIFORM GRAPH PARTITIONING and the TRAVELLING SALESMAN are in general only exact for $k \geq n - 1$. Furthermore, the neighbourhood graphs defined in the three examples are all strongly connected, implying that each solution can be reached from any other one in a finite sequence of $k$-exchanges. This is an important property in the analysis of the convergence of some algorithms such as for instance simulated annealing. The exchange neighbourhoods for UNIFORM GRAPH PARTITIONING and TRAVELLING SALESMAN given above are attributed to Kernighan & Lin [1970] and Lin & Kernighan [1973], respectively.

From the simple local search scheme of Figure 2, one can distinguish the following three basic steps in a local search algorithm: generation of a start solution, generation of a neighbour solution, and calculation of cost values.

For the examples discussed above these steps involve simple computations and efficient algorithms can be readily obtained. This is however not the case in general. In some cases efficient algorithms may require intricate computations or they even may not exist. For instance, finding feasible start solutions for VEHICLE ROUTING WITH TIME WINDOWS is NP-hard [Van der Bruggen, Lenstra, Savelsbergh & Schuur, 1991]. Also cost calculations may be quite difficult. For instance in VLSI design, placement is a well-known layout problem in which a set of modules is to be placed on a two-dimensional Euclidean map such that a weighted sum between the area of the surrounding box and the total length of the wires connecting the modules is minimal. Computing the wire length requires a routing model and for some models wire length computations are intractable, e.g., for the Steiner tree routing model [Sahni & Bhatt, 1980]. To illustrate some of the complications that may arise in the neighbourhood computations we discuss JOB SHOP SCHEDULING [French, 1982].

**Definition 2.9 (JOB SHOP SCHEDULING)** Given a set $J$ of $n$ jobs, a set $M$ of $m$ machines and a set $O$ of $N$ operations. For each operation $v \in O$ we are given a job $J_v \in J$ to which it belongs, a machine $M_v \in M$ on which it must be processed and a processing time $t_v \in \mathbb{Z}^+$. Furthermore a binary relation $<$ is given that decomposes $O$ into chains. The problem is to find start times $s_v$, for all $v \in O$ such that the makespan defined as $\max_{v \in O} (s_v + t_v)$ is minimal subject to the following constraints. For all $v, w \in O$

\[(i) \quad s_v \geq 0,\]
\[(ii) \quad s_w \geq s_v + t_v \text{ if } v < w, \text{ and}\]
\[(iii) \quad s_w \geq s_v + t_v \text{ or } s_v \geq s_w + t_w \text{ if } M_v = M_w.\]

Constraint (i) of Definition 2.9 implies nonnegative start times, constraint (ii) accounts for the binary relation, i.e., if $v < w$, then $w$ cannot start before $v$ is finished, and constraint (iii) implies that no two operations can be processed on the same machine at the same time.

To apply local search we use the disjunctive graph representation introduced by Roy & Sussmann
[1964]. For this the set of operations is augmented with a dummy start operation preceding all other operations and a dummy stop operation succeeding all other operations. Next we define the weighted disjunctive graph $G = (O, A, E)$ with $A = \{(u, v) | u < v \}$ and $E = \{(u, v) | M_u = M_v \}$, i.e., the arc set corresponds to the binary relation and the edge set corresponds to a collection of cliques representing the operations that must be processed on the same machine. Furthermore, each node $u \in O$ is given a weight corresponding to its processing time $t_u$. A feasible solution uniquely corresponds to a directed acyclic graph $G' = (O, A, D)$ that is obtained from $G$ by giving each of the edges in $E$ an orientation; let the set of oriented edges of $E$ be denoted by $D$. Furthermore the makespan of such a solution is given by the longest path in $G'$, which can be computed in $O(|O|^2)$ time by the Bellman-Ford algorithm [Nemhauser & Wolsey, 1988]. Thus $S$ can be chosen as the set of all acyclic directed graphs $G'$ that can be obtained from $G$, and the cost function is given by the longest path in $G'$.

A simple $k$-exchange neighbourhood structure can be defined as follows. Let $i$ be a solution with acyclic directed graph $G'_i = (O, A, D_i)$ For all $i \in S$, $N^{(k)}(i)$ is given by the set of acyclic graphs $G'_j$ that can be obtained from $G'_i$ by reversing the direction of $k$ arcs $D_i$, such that $G'_j$ is acyclic. The best way of ensuring feasibility is to test a graph obtained by a $k$-exchange for its acyclicity. This can be done in $O(|E| + |D|)$ time by a simple depth-first search algorithm. It can be easily verified that the neighbourhood graph obtained in this way is strongly connected. For more details the reader is referred to Van Laarhoven, Aarts & Lenstra [1992].

### 2.3 More elaborate exchange neighbourhoods

A major point of concern in local search is the observation that the algorithms may get stuck in bad local optima. It is generally known that for small values of $k$ the simple exchange neighbourhoods can be easily explored but on average yield low quality solutions - see also Section 2.4. On the other hand, for large values of $k$, the exchange neighbourhoods on average yield better solutions but often at the cost of substantial amounts of running time. So, research has been concentrating on the construction of more intricate neighbourhoods that can be explored without compromising too much on computational effort. Below, we discuss a few achievements in this area.

One of the major achievements is the class of variable-depth search algorithms. These algorithms were first introduced by Kernighan & Lin [1970] for UNIFORM GRAPH PARTITIONING and by the same authors for TRAVELLING SALESMAN [Lin & Kernighan, 1973]. However, many other problems have been handled ever since with the same type of algorithms. Here we discuss UNIFORM GRAPH PARTITIONING, for the reason that this problem best illustrates the typical features of variable-depth search.

Consider UNIFORM GRAPH PARTITIONING; see Definition 2.7. Let $V_1, V_2$ be a uniform partition of $V$, then the gain $g(a, b)$ in cost that is obtained by exchanging the nodes $a \in V_1$ and $b \in V_2$ is given by

$$g(a, b) = \sum_{\{a, v\} \in E, \ v \in V_2 \setminus \{b\}} w_{\{a, v\}} - \sum_{\{a, u\} \in E, \ u \in V_1} w_{\{a, u\}}$$
Note that the gain can be both negative and positive.

The variable-depth search algorithm replaces a single 2-exchange by a well-chosen sequence of \( k \) 2-exchanges using the gain of an exchange to guide the search. Furthermore, the value of \( k \) may vary from iteration to iteration. For a given partition \((V_1, V_2)\) of \( V \) with \( |V_1| = |V_2| = n \), the variable-depth exchange neighbourhood structure defines a neighbouring solution by the following steps.

Step 1. Choose two nodes \( a \in V_1 \) and \( b \in V_2 \) such that \( g(a, b) \) is maximal, not necessarily positive.

Step 2. Update \( V_1 \) and \( V_2 \), i.e. effectuate a tentative exchange of \( a \) and \( b \).

Step 3. Repeat Step 2 and Step 3 \( n \) times, where a node cannot be chosen to be exchanged if it has been exchanged in one of the previous iterations of Step 2 and Step 3.

The sequence of 2-exchanges obtained in this way defines a sequence \( g_1, g_2, \ldots, g_n \) of gains \( g_i \), where \( g_i \) corresponds to the exchange of the nodes \( a_i \in V_1 \) and \( b_i \in V_2 \) in the \( i \)-th iteration. The total gain after \( k \) exchanges equals

\[
G(k) = \sum_{i=1}^{k} g_i, \quad 0 \leq k \leq n.
\]

The corresponding sets of nodes in \( V_1 \) and \( V_2 \) whose exchanges lead to this gain are given by \( V_1^{(k)} = \{a_1, a_2, \ldots, a_k\} \) and \( V_2^{(k)} = \{b_1, b_2, \ldots, b_k\} \), respectively. Note that for \( k = n \) we have that \( V_1^{(n)} = V_1 \) and \( V_2^{(n)} = V_2 \), i.e., \( V_1 \) and \( V_2 \) have been entirely interchanged, resulting in a total gain \( G(n) = 0 \).

Step 4. Choose the value of \( k \) for which \( G(k) \) is maximal. If \( G(k) > 0 \), then the neighbouring solution is given by the partition that is obtained from \((V_1, V_2)\) by effectuating the definite exchange of \( V_1^{(k)} \) and \( V_2^{(k)} \). If \( G(k) \leq 0 \), then \((V_1, V_2)\) has no neighbouring solutions.

Note that in the neigbourhood structure given above each solution has at most one neighbouring solution.

Clearly, the basic idea behind the variable-depth exchange presented above is to allow unfavourable 2-exchanges in the sequence to eventually obtain a favourable \( k \)-exchange without exhaustive search of the \( k \)-exchange neighbourhood. If a favourable \( k \)-exchange exists it is always found by the above procedure. So, the final solution obtained by repeating the procedure until no favourable \( k \)-exchange is found anymore, is locally optimal with respect to the applied neighbourhood structure.

Essentially, variable-depth search algorithms allow the search to go to an unbounded depth avoiding the exponential growth of the simple exchange algorithms. The idea of variable-depth search
can also be applied to other problems such as TRAVELLING SALESMAN, JOB SHOP SCHEDULING or more general versions of partitioning and clustering problems. However, in these applications things are slightly more complicated since there generally is no natural bound on the number of simple exchanges as in the case of UNIFORM GRAPH PARTITIONING. One usually resorts to a heuristically chosen bound on the depth of the search. Furthermore, the gain measure might be more intricate then the one used for UNIFORM GRAPH PARTITIONING. Both these elements are present in the variable-depth search algorithm for TRAVELLING SALESMAN proposed by Lin and Kernighan and the details can be found in their paper [Lin & Kernighan, 1973].

Another approach to the modification of simple exchange algorithms is given by the reduction of the exchange neighbourhoods such that the efficiency of the algorithms is improved without affecting their effectiveness. Examples of such modifications for UNIFORM GRAPH PARTITIONING are given by Fiduccia & Mattheyses [1982], who propose to split up a 2-exchange into two 1-exchanges, by first selecting the best node to move from $V_1$ to $V_2$, and then the best node to move from $V_2$ to $V_1$, excluding the node that has already been exchanged. This idea was further refined by Dunlop & Kernighan [1985], who also made it applicable to the placement of standard cells in VLSI layout.

For TRAVELLING SALESMAN a number of elegant modifications have been proposed by Reinelt [1992]. He proposes to reduce the standard quadratic 2-exchange neighbourhoods to a linear 2-exchange neighbourhood by considering only exchanges among cities that are close in some sense. To quantify the notion of closeness, he proposes to use nearest neighbour graphs or Delaunay graphs. Similar approaches have been presented by Bentley [1990] and Johnson [1990b], who in addition used sophisticated data structures to support efficient implementations of 2-exchanges based on the concept of $k$-dimensional trees.

We finally briefly discuss the use of neighbourhood reduction techniques for JOB SHOP SCHEDULING; see Definition 2.9. Van Laarhoven, Aarts & Lenstra [1992] have proved the following results. Let $G$ be a directed weighted graph representing a feasible solution of an instance of JOB SHOP SCHEDULING. Then

- reversing an arbitrary arc on a longest path in $G$ always leads to an acyclic graph, and
- reversing an arc not belonging to a longest path in $G$ cannot reduce the length of the longest path in $G$.

So, evidently, the neighbourhoods in JOB SHOP SCHEDULING can be restricted to exchanges of arcs on a longest path. Note that this not only leads to a substantial reduction of the size of the neighbourhoods, but also renders the feasibility test superfluous. Furthermore, the same authors prove the following interesting property.

Let $G'_t$ be a directed weighted graph representing a feasible solution of an instance of JOB SHOP SCHEDULING. Then there exists a finite sequence of graphs $G'_{t_0}, G'_{t_1}, \ldots, G'_{t_m}$, each corresponding to a feasible solution, such that $G'_{t_0} = G'_t$ and $G'_{t_m}$ is optimal, and $G'_{t_{k+1}}$ is obtained from $G'_{t_k}$ for $k = 1, 2, \ldots, m$ by exchanging one edge on a longest path. Thus the neighbourhood graph is weakly optimally connected, i.e., there is a path from every node to an optimal node.
The neighbourhood structure can be further refined by allowing only specific operations to be exchanged. For instance, Matsuo, Suh & Sullivan [1988] propose a neighbourhood consisting of solutions in which not only the reversed arc \((u, v)\) must be on a longest path but in which also at least one of the arcs \((p(u), u)\) or \((v, s(v))\) may not be on a longest path, where \(p(u)\) and \(s(v)\) denote the predecessor and successor operations of \(u\) and \(v\), respectively. This condition further reduces the size of the neighbourhoods and causes the corresponding neighbourhood graph to be no longer weakly optimally connected.

### 2.4 Analysis and complexity

Unless a local search algorithm employs an exact neighbourhood structure, it is generally not possible to give non-trivial bounds on the amount by which the cost of a local optimum deviates from the optimal cost. Furthermore, for the general local search algorithm of Figure 2 no trivial bounds on the time complexity can be derived. However, in practice, many examples of local search algorithms are known that converge quickly and find high quality solutions. In this section we discuss the performance of local search in more detail, concentrating on both practical and theoretical aspects.

In the performance analysis of local search algorithms one may distinguish between average case and worst case, either theoretical or empirical, where the empirical worst-case analysis is of less significance. The performance of an approximation algorithm can be quantified by its running time and its deviation. The running time is usually given by the number of CPU seconds the algorithm requires to find a final solution. The deviation is given by the relative deviation in cost value of the final solution with respect to the optimal value or some lower bound on this value.

**Empirical results**

Over the years many results have been published on the empirical performance analysis of local search algorithms and a general conclusion is that local search algorithms can find good solutions within low order polynomial running times. Below, we elaborate on this issue in more detail.

For UNIFORM GRAPH PARTITIONING, Kernighan & Lin [1970] observed that the variable-depth search algorithm found solutions within a few percent of the optimum, using an average case running time of \(O(n^{2.4})\), where \(2n\) denotes the number of nodes. Furthermore, they showed that the probability of finding optimal solutions in the range they considered was about \(2^{-n/30}\). The implementation of Fiduccia & Mattheyses reduced the average case running time to \(O(n \log n)\), without substantial loss of quality of the final solutions. More about the performance of local search algorithms for UNIFORM GRAPH PARTITIONING can be found in [Johnson, Aragon, McGeoch & Schevon, 1989].

For JOB SHOP SCHEDULING Van Laarhoven, Aarts & Lenstra [1992] found that simple exchange algorithms could find solutions within 15% of the optimum with large probability. These results are not as good as those for UNIFORM GRAPH PARTITIONING or TRAVELLING SALESMAN, but they are still considerably better than those obtained with classical constructive heuristics. In a recent study, Applegate & Cook [1991] proposed a number of heuristics, not applying local search techniques, that yielded better results, both in terms of cost and running time. However,
it was possible to improve on these results with other local search based algorithms such as simulated annealing and genetic algorithms; see Aarts, Van Laarhoven, Lenstra & Ulder [1991]. First attempts to design effective variable-depth search algorithms for JOB SHOP SCHEDULING have failed so far. This is probably due to the fact that no effective exchange neighbourhood could be found for the underlying simple approach. All the known neighbourhoods are to small to allow sufficiently large steps to avoid getting trapped in poor local minima.

TRAVELLING SALESMAN is probably the most studied problem for local search. In their first paper Lin & Kernighan [1973] reported on an extensive numerical study. They found that 3-exchange neighbourhoods performed better than 2-exchange neighbourhoods but that the use of 4-exchange neighbourhoods did not sufficiently improve upon 3-exchange neighbourhoods to justify the additional running times. The probability of solutions to be optimal obtained by 3-exchanges was found to be equal to $2^{-n/10}$, where $n$ denotes the number of cities. Furthermore they found that variable-depth search performed very well: final solutions within a few percent of the optimum are found within relatively small running times. Average case running times are reported of $O(n^{2.9})$.

More recently Johnson and co-workers reported on an extensive numerical performance study of local search for TRAVELLING SALESMAN [Bentley, 1990; Johnson, 1990b; Bentley, Johnson, McGeoch & Rothberg, 1990]. This study uses randomly generated instances with sizes from $10^2$ up to $10^6$ cities. Their goal is to develop the best possible implementations of various local search algorithms for TRAVELLING SALESMAN and compare their performance with that of other heuristics, predominantly tour construction methods. Johnson [1991], reports a gain in speed of more than a factor 1000 for the variable-depth search algorithm compared to the first implementations of Lin & Kernighan. From this, a factor 40 is due to an increase in computational power of modern computers; a factor 25 is due to the use of more sophisticated data structures. Table 1 one shows a typical example of the obtained results.

<table>
<thead>
<tr>
<th></th>
<th>running time (sec.)</th>
<th>deviation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>10^2</td>
<td>10^3</td>
</tr>
<tr>
<td>greedy</td>
<td>0.2</td>
<td>1.6</td>
</tr>
<tr>
<td>farthest ins.</td>
<td>0.3</td>
<td>5.6</td>
</tr>
<tr>
<td>Christofides</td>
<td>2.8</td>
<td>992.9</td>
</tr>
<tr>
<td>2-exchange</td>
<td>0.4</td>
<td>5.0</td>
</tr>
<tr>
<td>3-exchange</td>
<td>0.5</td>
<td>6.6</td>
</tr>
<tr>
<td>var. depth</td>
<td>1.1</td>
<td>16.7</td>
</tr>
</tbody>
</table>

Table 1: Running times in seconds on a VAX.8550 and deviation in percentages relative to the Held-Karp lower bound [Held & Karp, 1970; 1971] of three tour construction methods, i.e., a greedy algorithm, farthest insertion and the Christofides algorithm and three local search algorithms for randomly generated problem instances of different sizes $n$. The table is taken from [Johnson, 1990a].
An important feature of this work is that the comparison is carried out for very large instances. Bentley [1990] has shown that there is a hidden $\Omega(n^{1.74})$ term in the running time of the exchange algorithms due to the growth of the relative length of the subpath that needs to be reversed after each exchange. It was argued that this term is significant at $n = 10^5$, while it is dominant at $n = 10^6$. This implies that for large instance sizes, the 2- and 3-exchange algorithms have empirical average case running times of the same order. The empirical average case running times of the variable-depth search algorithm for TRAVELLING SALESMAN is also slightly less then quadratic. Another important observation from this work is that for the larger instances the local search algorithms are superior to the tour construction heuristics; the deviations are substantially smaller, whereas the running times, at least for the 2- and 3-exchange algorithms, stay within the same order of magnitude. If running time is of no concern, then the variable-depth search algorithm is to be preferred. If ease of implementation plays a role, then one might choose for the simple exchange algorithms, given the greater programming complexity of the variable-depth search algorithm.

In another important study, Reinelt [1992] investigates local search algorithms for TRAVELLING SALESMAN on large real world instances. This approach is based on variable-depth search applying a limited 2-exchange neighbourhood structure as basic component; see previous section. Essential to the approach is the use of data structures based on a search tree, which keeps track of a list of intervals that correspond to subpaths that are not affected by applying the sequence of 2-exchanges on a given solution. This idea, which is due to Applegate, Chvátal & Cook [1990] considerably speeds up the algorithm. Compared to the results obtained by Johnson and co-workers, it seems that solutions can be obtained of about equal quality but in shorter running times. This conclusion however should be treated with caution, since the implementations of the variable-depth search algorithm used by both authors are quite different. Furthermore, there is an ongoing discussion about the difference between the use of randomly generated instances and real world instances. The latter class seems to be slightly more difficult to handle.

Theoretical results
In the previous section we have argued that local search algorithms on average behave quite well in practice; good solutions can be found in low-order polynomial running times. In this section we discuss a number of theoretical results showing the limitations of local search, at least from a worst-case point of view. It has been known for quite some time that there are problem instances for which local search algorithms may require an exponential number of steps to converge to a final solution. Probably the best known example is the simplex algorithm for linear programming [Klee & Minty, 1972]. However the simplex algorithm uses an exact neighbourhood and one might expect that just finding a locally optimal solution and not a globally optimal one is more easy. On the other hand, one might expect that it should be possible to find locally optimal solutions in polynomial time by other more sophisticated algorithms than local search. Unfortunately, this is not true in many cases. The literature presents a number of “bad examples”. Most of them are obtained for TRAVELLING SALESMAN, but there is no reason to expect that other NP-hard problems exhibit more favourable characteristics. Indeed, it has been possible to formulate a general complexity theory for local search that captures the characteristics for a broad class of problem. Before we discuss this theory in more detail, we mention some of the bad examples.
Weiner, Savage & Baghi [1976] were the first who proved that exact neighbourhoods for TRAVELING SALESMAN must have exponential sizes, provided that the neighbourhoods do not depend on the instances. Papadimitriou & Steiglitz [1977] have proved that, for general instances of TRAVELLING SALESMAN, no performance guarantee exists to find solutions with a constant deviation, and furthermore that neighbourhoods of polynomial size cannot be exact. Furthermore, they have constructed examples of instances with one optimal solution and exponentially many second best solutions that are locally optimal with respect to \( k \)-exchange neighbourhoods, for \( k < 3/8n \), while their costs are arbitrarily far from optimal. [Papadimitriou & Steiglitz, 1978].

Lueker [1975] has constructed instances and start solutions of TRAVELLING SALESMAN for which 2-exchange algorithms require an exponential number of steps. Examples of instances with bad worst-case behaviour for the independent set problem are given by Rodl & Tovey [1987].

Other investigations have concerned the theoretical average case behaviour of local search. Kern [1989] showed with a probabilistic analysis that the 2-exchange algorithm for Euclidean instances of TRAVELLING SALESMAN with \( n \) points in the unit square requires an average case running time \( T \) that is bounded by

\[
T \leq \frac{\sqrt{2}n^3}{d^*},
\]

where

\[
d^* = \min_{i,j \in S} \{|f(i) - f(j)|, i \in N(i) \text{ and } f(i) \neq f(j)\}.
\]

For arbitrary instances \( d^* \) may be arbitrarily small. However, if the points are uniformly distributed in \([0,1]^2\), then there exists a small number \( c > 0 \) such that \( d^* > n^{-15} \) with probability \( 1 - c/n \), implying that the algorithm stops after \( O(n^{18}) \) steps with probability greater than \( 1 - c/n \).

Other probabilistic models for local search have been studied by Tovey [1985; 1986]. In these studies artificial problems are considered with special classes of neighbourhood graphs with regular structures, e.g., the hypercube. The cost function for these problems is chosen to induce an orientation on the regular structure. Different probability distributions on the orientations are considered and for some cases low order polynomial average case running times are proved.

Recently, Johnson, Papadimitriou & Yannakakis [1988] formalized the question of the worst-case behaviour of local search algorithms by introducing a new complexity class. This new class and the underlying theoretical framework have proved a valuable tool in the analysis of local search. Below, we discuss this framework in more detail. But before we do so, we briefly summarize the basic elements of the more "classical" complexity theory.

In the mid sixties, Edmonds [1965] proposed to let the distinction between easy and hard problems coincide with the distinction between problems that can be solved efficiently, i.e., within running times bounded by a polynomial in the size of the instances, and those that are intractable, requiring running times that cannot be bounded by a polynomial in the instance size. Developments in theoretical computer science in the early seventies have led to a better understanding of this distinction. A class of decision problems was introduced which have a "yes" or "no" answer. Furthermore, for each problem instance a certificate property was introduced which can be used by a mathematical object called a Turing Machine [Turing, 1936]
to check the correctness of a "yes"-instance [Cook, 1972; Karp, 1972; Garey & Johnson, 1979].

Informally speaking this has led to the distinction between two classes, i.e., the class $P$ of decision problems that are solvable in polynomial time, and the class $NP$ of decision problems for which each "yes"-instance has a certificate that can be checked in polynomial time or, more loosely speaking, decision problems for which the correctness of each "yes"-instance can be checked in polynomial time. Clearly, $P$ is a subset of $NP$. Furthermore, a class $NPC \subset NP$ was conjectured of NP-complete problems, i.e., those problems in $NP$ that generalize all others and thus can be seen as the hardest ones. It is generally assumed that no algorithm can exist that solves each instance of an NP-complete problem using a running time that can be polynomially bounded in the size of the instance. However, if such an efficient algorithm would be found for one of these problems, then there must exist an efficient algorithm for each problem in $NP$, implying that $P = NPC$. To prove NP-completeness, the notion of reducibility has been introduced. A problem $\Pi$ is said to be polynomially reducible to a problem $\Pi'$ if there exists a polynomially computable function that maps each instance of $\Pi$ to an instance of $\Pi'$ such that a solution to an instance in $\Pi$ coincides with a solution to an instance in $\Pi'$. Furthermore, a generic problem called SATISFIABILITY exists to which all problems in $NP$ can be polynomially reduced. The transitivity property of the reducibility then can be used to show that a problem is in $NPC$ by (i) proving that it is in $NP$ and (ii) constructing a polynomial-time reduction from a problem that is NP-complete to the one at hand. Many problems have been proved NP-complete. For an extensive listing of many of these problems and a detailed introduction to the theory of NP-completeness, the reader is referred to [Garey & Johnson, 1979].

We now return to the complexity of local search. Johnson, Papadimitriou & Yannakakis [1988] introduced the class $PLS$ of Polynomial-time Local Search problems. To this end, a local search problem $\Pi_{LS}$ is formalized as a set of instances $x$, where each instance is characterized by a 3-tuple $(S_x, f_x, N_x)$, where $S_x$ is the set of solutions, $f_x$ a cost function and $N_x$ a neighbourhood structure as defined in the beginning of this section. The question is to find a local optimum with respect to $N_x$. A PLS problem is defined as follows.

**Definition 2.10** (polynomial-time local search problem) A local search problem $\Pi_{LS}$ is a PLS problem if there exist three polynomially computable functions $F_a$, $F_b$ and $F_c$ such that the following holds.

- On input $x \in \Pi_{LS}$, $F_a$ returns a start solution $i \in S_x$.
- On input $x \in \Pi_{LS}$ and $i \in S_x$, $F_b$ returns the cost value $f_x(i)$.
- On input $x \in \Pi_{LS}$ and $i \in S_x$, $F_c$ returns $x$ if $x$ is locally optimal with respect to $N_x$, and if not it returns a neighbouring solution with better cost.

In their original paper Johnson, Papadimitriou & Yannakakis [1988] made a distinction between feasible and infeasible solutions, which is omitted here for reasons of simplicity. Informally stated, $PLS$ defines the class of local search problems for which local optimality can be verified in polynomial time. Next there is the class $PLSC$ of PLS-complete problems that generalize all other problems in $PLS$. So, the problems in $PLSC$ are the hardest ones in $PLS$ and, moreover, solving them may take superpolynomial running times. Finally, if one of them can be shown to be
solvable in polynomial time, then all the others can. To prove that a problem is PLS-complete, again one must show that (i) it is in $P_{LS}$ and that (ii) there exists a known PLS-complete problem that can be PLS-reduced to the problem at hand. The notion of PLS-reducibility is defined as follows.

**Definition 2.11** (polynomial-time local search reduction) A problem $\Pi_{LS} \in P_{LS}$ is PLS-reducible to another problem $\Pi'_{LS} \in P_{LS}$ if there exist polynomially computable functions $h$ and $g$ such that the following holds.

- $g$ Maps instances of $\Pi_{LS}$ onto instances of $\Pi'_{LS}$.
- For all $x \in \Pi_{LS}$, $h$ maps pairs of the form $(\text{solution of } g(x), x)$ onto solutions of $x$.
- For all $x \in \Pi_{LS}$, if $i$ is locally optimal for $g(x)$, then $h(i, x)$ is locally optimal for $x$.

So, a PLS-reduction not only maps instances of one problem to those of another problem, as in the case of "classical reductions", but also maps the entire structure of the neighbourhood graph of one problem to that of another problem such that the topology of both structures with respect to local optimality is the same. The generic PLS-complete problem is FLIP and is defined as follows.

**Definition 2.12** (FLIP) Given a feedback-free Boolean circuit of AND, OR and NOR gates with $m$ inputs and $n$ outputs. For each binary input vector $x \in \{0,1\}^m$ a binary output vector $y(x) \in \{a, l\}^n$ is computed. Furthermore, let

- $S = \{x \mid x \in \{0,1\}^m\}$,
- $f(x) = \sum_{j=1}^{n} y_j(x)2^{(j-1)}$, for all $x \in S$,
- $N(x) = \{z \in S \mid d(x,z) \leq 1\}$, for all $x \in S$,
  where $d(x,z)$ denotes the number of positions at which $x$ and $z$ differ, and,
- $\forall x \in S$:
  $F_a(x) = (1, \ldots, 1)$,
  $F_b(x) = f(x)$, and
  $F_c(x) = \begin{cases} x & \text{if } f(x) = \min_{v \in N(x)} f(v) \\ z & \text{otherwise,} \end{cases}$
  where $z \in N(x)$ and $f(z) = \min_{v \in N(x)} f(v)$.

Find a local minimum of $f$ with respect to $N$.

Since its introduction, the class $P_{LS}$ has received considerable attention and a number of local search problems have been proved PLS-complete. Important contributions to this area have been made by Krentel [1989; 1990] and Schäffer & Yannakakis [1991]. Below, we briefly discuss some of their results.
The precise positioning of the class \( \mathcal{P}L\mathcal{S} \) is still open. It lies somewhere between \( \mathcal{P} \) and \( \mathcal{N}\mathcal{P} \), respectively, i.e., \( \mathcal{P} \subseteq \mathcal{P}L\mathcal{S} \subseteq \mathcal{N}\mathcal{P} \). It is however assumed that PLS problems cannot be NP-hard since this would imply that \( \mathcal{N}\mathcal{P} = \text{co-}\mathcal{N}\mathcal{P} \), which is unlikely [Johnson, Papadimitriou & Yannakakis, 1988]. On the one hand, LINEAR PROGRAMMING with the simplex neighbourhood is in \( \mathcal{P}L\mathcal{S} \), whereas LINEAR PROGRAMMING itself is in \( \mathcal{P} \). On the other hand, to show that all problems in \( \mathcal{P}L\mathcal{S} \) can be solved in polynomial time would require the construction of a general method for local search problems that would be at least as intricate as the ellipsoid method, which is quite a difficult exercise. Krentel [1989] argues that PLS-completeness is the normal behaviour of local search variants of NP-complete problems and he conjectures that more than half of the problems mentioned in Garey & Johnson [1979] have PLS-complete variants.

So far, a number of local search problems have been proved PLS-complete. Many of the PLS-reductions presented in the literature require delicate and complex constructions and in most cases NP-completeness reductions are of little help. The following problems are known to be PLS-complete: UNIFORM GRAPH PARTITIONING and TRAVELLING SALESMAN with the variable-depth search neighbourhoods; UNIFORM GRAPH PARTITIONING with the \( k \)-exchange neighbourhoods for arbitrary \( k \geq 2 \) and with the Fiduccia & Mattheyses neighbourhood; SATISFIABILITY and 2SAT with the FLIP neighbourhood; MAXCUT with the \( k \)-exchange neighbourhoods, and STABLE CONFIGURATION with the FLIP neighbourhood; for the definition of the problems mentioned here, see for instance [Schäffer & Yannakakis, 1991]. STABLE CONFIGURATION is related to neural networks and its PLS-completeness generalizes previously known complexity results on the running time required to reach stable configurations in neural networks. We return to this problem in Section 4. For TRAVELLING SALESMAN there exists an interesting open problem with respect to its PLS-complexity. Krentel [1989] has shown that TRAVELLING SALESMAN with \( k \)-exchange neighbourhoods is PLS-complete for sufficiently large values of \( k \); a conservative estimate is for \( k \)-values between \( 10^3 \) and \( 10^4 \). It remains an open problem to determine whether the same holds for 2-exchange and 3-exchange neighbourhoods.

Several of the above mentioned problems are shown to be P-complete in the unweighted case. From intuitive arguments this is quite obvious since in each step of the local search the cost value is lowered by a finite amount, whereas the range of values is polynomially bounded in the problem size.

Furthermore, there is an issue in the theory of PLS-completeness. In a PLS-completeness proof, one must solve the subproblem of verifying local optimality of a given solution. With respect to this aspects the known PLS-complete problems can be divided into two classes, i.e., those for which the verification subproblem is P-complete and those for which it is in LOGSPACE; for definitions, see Garey & Johnson [1979]. This indicates that P-completeness is not a necessary condition for PLS-completeness as was conjectured by Johnson, Papadimitriou & Yannakakis [1988]. Furthermore, it implies that the fact that a verification subproblem is in LOGSPACE is not a necessary and sufficient condition for finding locally optimal solutions of the corresponding PLS-problem.

Thus, problems can be PLS-complete despite the fact that the verification subproblem does not use the full polynomial time that is allowed. An example is UNIFORM GRAPH PARTI-
TIONING. For both the 2-exchange and the variable-depth neighbourhoods the problem is PLS-complete. However the verification subproblem for the 2-exchange neighbourhood is in \textsc{LOGSPACE} whereas that for the variable-depth neighbourhood is P-complete [Schäffer & Yannakakis, 1991]. Another example is the PLS-complete problem \textsc{CNF FLIP}, for which Krentel [1990] showed that the verification subproblem is in \textsc{LOGSPACE}.

Finally we mention that the complexity results discussed above can also be applied to local search algorithms not following the strict improvement strategies of Figure 2. For instance, replacement of the deterministic improvement strategy by a randomized strategy, as in simulated annealing, does not change any of the complexity results discussed above.

2.5 Related topics

Most of the local search algorithms discussed above have the advantage of being \textit{generally applicable} and \textit{flexible}, i.e., only a specification of solutions, a cost function and a neighbourhood structure are required, which can be straightforwardly obtained for most problems. A disadvantage is that in a number of cases poor local optima are found. To avoid this disadvantage - while maintaining the paradigm of local search algorithms, i.e., iteration among neighbouring solutions - many people have been investigating the possibilities to broaden the scope of local search, for instance by allowing deteriorations in cost. This has led to a variety of algorithms. Two well-known examples are simulated annealing and tabu search. Simulated annealing [Kirkpatrick, Gelatt & Vecchi, 1983; Černý, 1985] is extensively discussed in Section 3.

Tabu search [Glover, 1989a; 1989b] is a meta-heuristic, which, to simplify it slightly, applies a neighbourhood search that is guided by a tabu list consisting of solutions that may not be selected in a later step of the algorithm. The tabu list is dynamically updated in the course of the algorithm's execution. In a way one might view tabu search as a local search algorithm applying a dynamic neighbourhood structure. In a recent paper, Glover [1991] indeed points out the relation between tabu search and the special class of local search algorithms called threshold algorithms; see also Section 3.1.

Tabu search has been applied to a large variety of problems and considerable successes have been reported on its performance; see for instance Hertz & De Werra [1991]. As already mentioned, tabu search is a meta-heuristic. More specifically this means that there is a number of parameters involved that must be instantiated before application. Unfortunately, there is little theoretical knowledge that guides the choice of these parameters, and users have to resort to the available practical experience. This makes it hard to give a mathematical overview of the algorithm and for that reason we do not consider it in more detail, despite the fact that it deserves more on the basis of its success in practice.

Multiple-runs approaches
A straightforward extension of local search would be to run a simple local search algorithm a number of times using different start solutions and keeping the best found solution to return eventually as the final solution. Several authors have investigated this approach, but no major successes have been reported; see for instance Aarts & Van Laarhoven [1989] for TRAVELLING SALESMAN and Aarts, Van Laarhoven, Lenstra & Ulder [1991] for JOB SHOP SCHEDULING.
Instead of restarting with an arbitrary initial solution, one could consider an approach that applies multiple runs of a local search algorithm combining the information obtained from these runs to select appropriate start solutions for subsequent runs. One such approach is **iterated local search** in which the start solutions of subsequent local searches are obtained by modifying the local optima of a previous run. An example for TRAVELLING SALESMAN is the **iterated Lin Kernighan** algorithm proposed by Johnson [1990a], which uses multiple runs of the variable-depth search algorithm of Lin & Kernighan with a random 4-exchange to obtain new start solutions. Johnson reports excellent results for this approach; solutions within 0.8% of the Held & Karp lower bound on random Euclidean instances with sizes from 100 to 10,000 cities. In fact, for TRAVELLING SALESMAN this is currently the best-known approximation algorithm, even when running times are taken into account.

Another approach to multiple runs is based on a theme borrowed from population genetics [Goldberg, 1989]. This has led to the class of **genetic local search algorithms**, which was first proposed by Mühlenbein, Gorges-Schleuter & Krämer [1988]. The general idea in these algorithms is best explained by the following scheme.

**Step 1. Initialize:** Construct an initial population of $n$ solutions.

**Step 2. Improve:** Use local search to replace the $n$ solutions in the population by $n$ local optima.

**Step 3. Recombine:** Augment the population by adding $m$ offspring solutions. The population size now equals $n + m$.

**Step 4. Improve:** Use local search to replace the $m$ offspring solutions by $m$ local optima.

**Step 5. Select:** Reduce the population to its original size by selecting $n$ solutions from the current population.

**Step 6. Evolute:** Repeat Steps 3-5 until a stop criterion is satisfied.

Evidently, **recombine** is an important step, since here one must try to take advantage of the fact that more than one local optimum is available, i.e., one must exploit the structure present in the available local optima.

The above scheme has been applied to several problems and good results have been reported; see for instance Mühlenbein, Gorges-Schleuter & Krämer [1988] and Ulder, Aarts, Bandelt, Van Laarhoven & Pesch [1990] for TRAVELLING SALESMAN and Aarts, Van Laarhoven, Lenstra & Ulder [1991] for JOB SHOP SCHEDULING. For all these applications genetic local search algorithms could find solutions comparable in quality to those obtained by existing approximation algorithms. Vaessens, Aarts & Van Lint [1990] have used genetic local search algorithms to improve upon the best known upper bounds in the literature for a number of instances of packing problems in code design; see also Aarts & Van Laarhoven [1992].

Finally we mention that Vaessens, Aarts & Lenstra [1992] have given a model that combines the majority of the existing local search algorithms into one algorithmic template. The various algorithms follow from the template by instantiating certain parameters and filling in specific
procedures. The generic features of the template are obtained by introducing several levels at which different neighbourhoods can be used and by the so-called hyper neighbourhood structures, which define the neighbouring solutions for a set of solutions, instead of a single solution as is done in the classical neighbourhood structures.
3 Simulated annealing

Simulated annealing plays a special role within the class of local search algorithms because of the following two reasons. Firstly, it has been shown rather successful in a broad range of applications, which has given the algorithm quite a reputation among practitioners. Secondly, due to its stochastic nature, it is possible to mathematically analyze the algorithm and to prove a number of interesting properties related to its asymptotic convergence, which makes the algorithm attractive to mathematicians. It should be noted that such an analysis is impracticable for deterministic local search algorithms. The emphasis of our presentation is clearly on the second issue, considering the scope of this book. However, where possible, we also mention some of the practical implications of the algorithm, so as to give the reader a feeling for its use in practice.

3.1 Threshold algorithms

As already mentioned, simulated annealing can be viewed as a randomized version of local search. To illustrate this we introduce the class of threshold algorithms. Threshold algorithms are obtained by replacing in the pseudocode of Figure 2, the lines

\[
\text{if } f(j) < f(i) \text{ then } i := j; \\
\text{until } f(j) \geq f(i), \text{ for all } j \in N(i);
\]

by the following lines (in the case of minimization)

\[
\text{if } f(j) - f(i) < t_k \text{ then } i := j; \\
k := k + 1; \\
\text{until some stop criterion is met.}
\]

Here \((t_k | k = 0, 1, 2, \ldots)\) denotes a sequence of thresholds, where the threshold \(t_k\) is used at iteration \(k\) of the local search algorithm. We can distinguish the following three types of threshold algorithms depending on the nature of the thresholds.

- **Iterative improvement**
  
  \[ t_k = 0, \quad k = 0, 1, 2, \ldots \]

  Clearly, this is just a reformulation of the local search procedure given in Figure 2.

- **Threshold accepting**
  
  \[ t_k = c_k, \quad k = 0, 1, 2, \ldots, \]

  where \(c_k \geq 0, c_{k+1} \leq c_k\), and \(\lim_{k \to \infty} c_k = 0\).

Threshold accepting uses a non-increasing sequence of deterministic thresholds. Due to the use of positive thresholds, neighbouring solutions with larger costs are accepted in a limited way. In the course of the minimization, the threshold values become equal to 0, in which case only improvements are accepted. Threshold accepting was introduced by Dueck & Scheuer.
[1990] as a deterministic version of simulated annealing. For several applications it has been shown quite successful. One of the problems, however, is to determine appropriate values for the thresholds. This problem has not been satisfactorily resolved so far.

Moreover, in contrast to simulated annealing, no general asymptotic convergence guarantee on the performance of the algorithm can be given. Althöfer & Koschnick [1991] have related some convergence properties of threshold accepting to those of simulated annealing. The proofs of the convergence results are not constructive. They make use of the fact that in some sense simulated annealing belongs to the convex hull of threshold accepting. For instance, one of their propositions states that if there is a finite sequence of thresholds for simulated annealing, leading to an optimal solution with probability 1 - ε for some ε > 0, then there also exists a finite sequence of thresholds for threshold accepting leading to an optimal solution with probability 1 - ε. Furthermore, they give a simple example for which suboptimal solutions can be reached with threshold sequences of arbitrary length, demonstrating that even asymptotically the algorithm can get trapped in local minima.

- **Simulated annealing**

\[ t_k = \text{a positive stochastic variable with } \mathbb{E}(t_k) = c_k \in \mathbb{R}^+, \quad k = 0, 1, 2, \ldots \]

Furthermore, the \( t_k \)'s are given by a probability distribution function \( F_{c_k} \) over \( \mathbb{R}^+ \).

In other words, simulated annealing uses randomized thresholds with values between zero and infinity and the probability of a threshold \( t_k \) being equal to \( y \in \mathbb{R}^+ \) is given by \( \mathbb{P}_{c_k} \{ t_k \leq y \} = F_{c_k}(y) \). This implies that each neighbouring solution is accepted with a finite probability to replace the current solution. In practice the function \( F \) is chosen such that solutions corresponding to large increases in cost have a small probability of being accepted, whereas solutions corresponding to small increases in cost have a larger probability of being accepted. For instance, the original simulated annealing version of Kirkpatrick, Gellatt & Vecchi [1983] and Černý [1985] uses \( F_{c_k} = \mathcal{E}(1/c_k) \), i.e., the exponential distribution with parameter \( 1/c_k \). This choice is identical to the following **acceptance criterion**. For any two solutions \( i, j \in S \) the probability of accepting \( j \) from \( i \) at the \( k^{\text{th}} \) iteration is given by

\[
\mathbb{P}_{c_k} \{ \text{accept } j \} = \begin{cases} 1 & \text{if } f(j) \leq f(i) \\ \exp \left( \frac{f(i) - f(j)}{c_k} \right) & \text{if } f(j) > f(i). \end{cases}
\]

The parameter \( c_k \) is used in the simulated annealing algorithm as a **control parameter**.

The origin of simulated annealing and the choice of the acceptance criterion lies in the physical annealing process. In **condensed matter physics**, annealing is a thermal process for obtaining low energy states of a solid in a heat bath. The process contains the following two steps: firstly, the temperature of the heat bath is increased to a maximum value at which the solid melts; secondly, the temperature is carefully decreased until the particles of the melted solid arrange themselves in the ground state of the solid. In the liquid phase all particles of the solid arrange themselves randomly. In the ground state the particles are arranged in a highly structured lattice and the energy of the system is minimal.
The physical annealing process can be modelled successfully by computer simulation methods based on Monte Carlo techniques. An introductory overview of the use of these techniques in statistical physics is given by Binder [1978]. Here, we discuss one of the early techniques proposed by Metropolis, Rosenbluth, Rosenbluth, Teller & Teller [1953], who gave a simple algorithm for simulating the evolution of a solid in a heat bath to thermal equilibrium. Their algorithm is based on Monte Carlo techniques and generates a sequence of states of the solid in the following way. Given a current state \( i \) of the solid with energy \( E_i \), then a subsequent state \( j \) is generated by applying a perturbation mechanism which transforms the current state into a next state by a small distortion, for instance by displacement of a particle. The energy of the next state is \( E_j \). If the energy difference, \( E_j - E_i \), is less than or equal to 0, the state \( j \) is accepted as the current state. If the energy difference is greater than 0, the state \( j \) is accepted with a certain probability which is given by

\[
\exp \left( \frac{E_i - E_j}{k_B T} \right),
\]

where \( T \) denotes the temperature of the heat bath and \( k_B \) a physical constant known as the Boltzmann constant. The acceptance rule described above is known as the Metropolis criterion; the corresponding algorithm as the Metropolis algorithm.

If the lowering of the temperature is done sufficiently slowly, the solid can reach thermal equilibrium at each temperature, which is characterized by the Boltzmann distribution. This distribution gives the probability of the solid being in a state \( i \) with energy \( E_i \) at temperature \( T \), and is given by

\[
\Pi_T(X = i) = \frac{\exp(-E_i/k_B T)}{\sum_j \exp(-E_j/k_B T)},
\]

where \( X \) is a stochastic variable denoting the current state of the solid and the summation extends over all possible states. As we will see in the following sections, the Boltzmann distribution plays an essential role in the analysis of the simulated annealing algorithm.

Returning to simulated annealing, we can apply the Metropolis criterion to generate a sequence of solutions of a combinatorial optimization problem. For this purpose we assume an analogy between a physical many-particle system and a combinatorial optimization problem based on the following equivalences.

- Solutions in a combinatorial optimization problem are equivalent to states of a physical system.
- The cost of a solution is equivalent to the energy of a state.

Next, we introduce a parameter which plays the role of the temperature. This parameter is the same as the control parameter used in (1).

A characteristic feature of simulated annealing is that, besides accepting improvements in cost, it also accepts to a limited extent deteriorations in cost. Initially, at large values of \( c \), large deteriorations will be accepted; as \( c \) decreases, only smaller deteriorations will be accepted and
finally, as the value of $c$ approaches 0, no deteriorations will be accepted at all. Furthermore, there is no limitation on the size of a deterioration with respect to its acceptance, as for instance in threshold accepting. In simulated annealing, arbitrarily large deteriorations are accepted with finite probability; for these deteriorations the acceptance probability is small however.

3.2 Markov modelling


Definition 3.1 Let $\mathcal{O}$ denote a set of possible outcomes of a sampling process. A Markov chain is a sequence of trials, where the probability of the outcome of a given trial depends only on the outcome of the previous trial. Let $X(k)$ be a stochastic variable denoting the outcome of the $k^{th}$ trial, then the transition probability at the $k^{th}$ trial for each pair of outcomes $i, j \in \mathcal{O}$ is defined as

$$P_{ij}(k) = P\{X(k) = j | X(k - 1) = i\}.$$ (2)

The matrix $P(k)$ whose elements are given by (2) is called the transition matrix. Furthermore, a Markov chain is called finite if the set of outcomes is finite. It is called inhomogeneous if the transition probabilities depend on the trial number $k$. If they do not depend on the trial number, the Markov chain is called homogeneous.

Let $a_i(k)$ denote the probability of an outcome $i \in \mathcal{O}$ at the $k^{th}$ trial, i.e.,

$$a_i(k) = P\{X(k) = i\}.$$ (3)

Then for all $i \in \mathcal{O}$, $a_i(k)$ is given as

$$a_i(k) = \sum_{l \in \mathcal{O}} a_l(k - 1) P_{il}(k).$$

Definition 3.2 An $n$-vector $x$ is called stochastic if its components $x_i$ satisfy the following conditions.

$$x_i \geq 0, \ i = 1, \ldots, n, \ \text{and} \ \sum_{i=1}^{n} x_i = 1.$$ 

An $n \times m$- matrix $X$ is called stochastic if its components $X_{ij}$ satisfy the following conditions.

$$X_{ij} \geq 0, \ i = 1, \ldots, n, \ j = 1, \ldots, m, \ \text{and} \ \sum_{j=1}^{m} X_{ij} = 1, \ i = 1, \ldots, n.$$ 

In the case of simulated annealing, a trial corresponds to a transition, and the set of outcomes is given by the finite set of solutions. Furthermore, the outcome of a trial only depends on the outcome of the previous trial. Consequently, we may safely apply the concept of finite Markov chains.
Definition 3.3 (transition probability) Let \((S, J)\) be an instance of a combinatorial optimization problem and \(N\) a neighbourhood structure. Then the transition probabilities for the simulated annealing algorithm are defined as

\[
\forall i, j \in S : \quad P_{ij}(k) = P_{ij}(c) = \begin{cases} 
G_{ij}(c)A_{ij}(c) & \text{if } i \neq j \\
1 - \sum_{l \in \mathcal{N}(i), l \neq i} G_{il}(c)A_{il}(c) & \text{if } i = j,
\end{cases}
\]

where \(c = c_k\), \(G_{ij}(c)\) denotes the generation probability, i.e., the probability of generating a solution \(j\) from a solution \(i\), and \(A_{ij}(c)\) denotes the acceptance probability, i.e., the probability of accepting a solution \(j\) that is generated from solution \(i\).

Note that the matrix \(P(c)\) of (4) is stochastic. The \(G_{ij}(c)\)'s and \(A_{ij}(c)\)'s of (4) are conditional probabilities, i.e., \(G_{ij}(c) = \mathbb{P}_c\{j \mid i\}\) and \(A_{ij}(c) = \mathbb{P}_c\{j \mid i, j\}\). The corresponding matrices \(G(c)\) and \(A(c)\) are the generation matrix and acceptance matrix, respectively, and need not be stochastic.

In the original version of simulated annealing the following probabilities are used.

Definition 3.4

- (generation probability)

\[
\forall i, j \in S : \quad G_{ij}(c) = G_{ij} = \frac{1}{\Theta} \chi(\mathcal{N}(i))(j),
\]

where \(\Theta = |\mathcal{N}(i)|\), for all \(i \in S\), and the characteristic function \(\chi(A')\) is a mapping \(\chi(A') : A \to \{0, 1\}\) for a given \(A' \subseteq A\), which is defined as: \(\chi(A')(a) = 1\) if \(a \in A'\) and \(\chi(A')(a) = 0\) otherwise.

- (acceptance probability)

\[
\forall i, j \in S : \quad A_{ij}(c_k) = \exp \left( -\frac{(f(j) - f(i))^+}{c_k} \right),
\]

where, for all \(a \in \mathbb{R}\), \(a^+ = a\) if \(a > 0\), and \(a^+ = 0\) otherwise.

Thus, the generation probabilities are chosen to be independent of the control parameter \(c_k\) and uniformly over the neighbourhoods \(\mathcal{N}(i)\), where it is assumed that all neighbourhoods are of equal size, i.e., \(|\mathcal{N}(i)| = \Theta\), for all \(i \in S\).

The above definitions can be used for virtually all combinatorial optimization problems, and close examination of the literature reveals that in most practical applications these definitions - or minor variations - are used indeed. It is however also possible to formulate a set of conditions ensuring asymptotic convergence for a more general class of acceptance and generation probabilities. We return to this subject later.

We now concentrate on the asymptotic convergence of simulated annealing. A simulated annealing algorithm finds an optimal solution with probability \(1\) if, after a possibly large number of trials, we have

\[
\mathbb{P}\{X(k) \in S^*\} = 1.
\]

26
In the following sections we show that under certain conditions the simulated annealing algorithm converges asymptotically to the set of optimal solutions, i.e.,

$$\lim_{k \to \infty} \mathbb{P}\{X(k) \in S^*\} = 1.$$ 

Stationarity of Markov chains

An essential property in the study of Markov chains is that of stationarity. Under certain conditions on the transition probabilities associated with a Markov chain there exists a unique stationary distribution; see Feller [1950] and Isaacson & Madsen [1976].

**Definition 3.5 (stationarity)** A stationary distribution of a finite homogeneous Markov chain with transition matrix $P$ on a set of outcomes $O$ is defined as the stochastic $|O|$-vector $q$, whose components are given by

$$q_i = \lim_{k \to \infty} \mathbb{P}\{X(k) = i | X(0) = j\}, \quad \text{for all } j \in O.$$ 

If such a stationary distribution $q$ exists we have $\lim_{k \to \infty} a_i(k) = q_i$, where $a_i(k)$ is given by (3). Furthermore, it follows directly from the definitions that $q^T = q^T P$. Thus, $q$ is the probability distribution of the outcomes after an infinite number of trials and the left eigenvector of $P$ with eigenvalue 1. Clearly, in the case of the simulated annealing algorithm, as $P$ depends on $c$, $q$ depends on $c$, i.e., $q = q(c)$.

Before we can prove the existence of the stationary distribution for the simulated annealing algorithm, we need the following definitions.

**Definition 3.6 (irreducibility)** A Markov chain with transition matrix $P$ on a set of outcomes $O$ is irreducible if for each pair of outcomes $i, j \in O$ there is a positive probability of reaching $j$ from $i$ in a finite number of trials, i.e.,

$$\forall i, j \in O \ \exists n \in \mathbb{Z}^+ : (P^n)_{ij} > 0.$$ 

**Definition 3.7 (aperiodicity)** A Markov chain with transition matrix $P$ is aperiodic if for each outcome $i \in O$ the greatest common divisor $\gcd(D_i) = 1$, where $D_i$ is the set of all integers $n > 0$, with

$$(P^n)_{ii} > 0.$$ 

The integer $\gcd(D_i)$ is called the period of $i$. Thus, aperiodicity requires all solutions to have period 1. As a corollary we have that for an irreducible Markov chain aperiodicity holds if

$$\exists j \in O : P_{jj} > 0. \quad (7)$$

We now come to the following important theorem; see Feller [1950] and Isaacson & Madsen [1976].

**Theorem 3.1** Let $P$ be the transition matrix associated with a finite homogeneous Markov chain on a set of outcomes $O$ and let the Markov chain be irreducible and aperiodic. Then there exists a unique stationary distribution $q$ whose components $q_i$ are uniquely determined by the following equation.

$$\sum_{j \in O} q_j P_{ji} = q_i, \quad \text{for all } i \in O.$$ 

27
As a corollary we have that any probability distribution \( q \), which is associated with a finite, irreducible and aperiodic homogeneous Markov chain and which satisfies the following equations
\[
q_i P_{ij} = q_j P_{ji}, \quad \text{for all } i, j \in \mathcal{O},
\](8)
is the unique stationary distribution in the sense mentioned in Theorem 3.1. The equations of (8) are called the *detailed balance equations* and a Markov chain for which they hold is called *reversible*.

### 3.3 The homogeneous model

We now can prove asymptotic convergence of simulated annealing based on the model in which the algorithm is viewed as a sequence of infinitely long Markov chains.

**Theorem 3.2** Let \((\mathcal{S}, f)\) be an instance of a combinatorial optimization problem, \(\mathcal{N}\) a neighbourhood structure, and \(P(c)\) the transition matrix of the homogeneous Markov chain associated with the simulated annealing algorithm defined by (4), (5), and (6). Furthermore, let the following condition be satisfied.

\[
\forall i, j \in \mathcal{S} \ 3p \geq 1, \ 3l_0, l_1, \ldots, l_p \in \mathcal{S},
\]

with \(l_0 = i, l_p = j\), and

\[
G_{l_k l_{k+1}} > 0, \quad k = 0, 1, \ldots, p - 1.
\]

Then the Markov chain has a stationary distribution \(q(c)\), whose components are given by
\[
q_i(c) = \frac{\exp(-f(i)/c)}{\sum_{j \in \mathcal{S}} \exp(-f(j)/c)}, \quad \text{for all } i \in \mathcal{S},
\]
and
\[
\lim_{c \downarrow 0} q_i(c) \overset{\text{def}}{=} q_i^* = \frac{1}{|\mathcal{S}^*|} \chi(\mathcal{S}^*)(i).
\](10)

The proof of the theorem follows directly from the previous results, i.e., the condition in the theorem ensures that the Markov chain is irreducible; see Definition 3.6; the transition probabilities given by (4), with (5) and (6) ensure aperiodicity through (7); see Definition 3.7; thus according to Theorem 3.1, there exists a unique stationary distribution; the correctness of the components of (9) follows directly from the detailed balance equations of (8), which completes the first part of the theorem. The second part follows directly from (9). The distribution given by (9) is the equivalent in the Markov chain analysis of simulated annealing of the Boltzmann distribution in the Monte Carlo simulations of the physical annealing process mentioned in Section 3.1. It is characteristic of simulated annealing and, as we will see below, it plays an important role in the analysis of the algorithm.
As a result of Theorem 3.2 we have that
\[
\lim_{c \to 0} \lim_{k \to \infty} \mathbb{P}_c\{X(k) \in S'\} = 1. \quad (11)
\]
This result reflects the basic property of the simulated annealing algorithm, i.e., the guarantee that the algorithm asymptotically finds an optimal solution. Furthermore, (11) expresses the characteristic of the homogeneous model for simulated annealing, viz., first take the limit of the homogeneous Markov chain for an infinite number of trials and next take the limit for the control parameter to zero. In the inhomogeneous model of Section 3.4 these two limits are combined to a single one.

The convergence properties discussed above strongly depend on the original choice of the transition probabilities of (4), (5), and (6). Several authors have investigated convergence properties for a broader class of probabilities. This has led to the following formulation.

**Theorem 3.3** Let \((S, f)\) be an instance of a combinatorial optimization problem, \(N\) a neighbourhood structure and \(P(c)\) the transition matrix of the homogeneous Markov chain associated with the simulated annealing algorithm defined by (4). Furthermore, let the following conditions hold.

\begin{align*}
(G1) & \quad \forall c > 0, \forall i, j \in S, \exists \eta \geq 1, \exists l_0, l_1, \ldots, l_p \in S, \text{with } l_0 = i, l_p = j, \\
& \quad \text{and } G_{i_k l_{k+1}} (c) > 0, \quad k = 0, 1, \ldots, p - 1, \\
(G2) & \quad \forall c > 0, \forall i, j \in S : \quad G_{ij} (c) = G_{ji} (c), \\
(A1) & \quad \forall c > 0, \forall i, j \in S : \quad A_{ij} (c) = 1 \quad \text{if } f(i) \geq f(j), \\
& \quad \quad \quad A_{ij} (c) \in (0, 1) \quad \text{if } f(i) < f(j), \\
(A2) & \quad \forall c > 0, \forall i, j, k \in S : \quad A_{ij} (c)A_{jk} (c)A_{ki} (c) = A_{ik} (c)A_{kj} (c)A_{ji} (c), \\
(A3) & \quad \forall i, j \in S \text{ with } f(i) < f(j) : \lim_{c \downarrow 0} A_{ij} (c) = 0.
\end{align*}

Then the Markov chain has a unique stationary distribution \(q(c)\), whose components are given by

\[
q_i (c) = \frac{1}{\sum_{j \in S} A_{ij} (c)/A_{ji} (c)}, \quad \text{for all } i \in S, \quad (12)
\]

and

\[
\lim_{c \downarrow 0} q_i (c) = q^*_i,
\]

where the \(q^*_i\) are given by (10).

Condition (G1) again ensures irreducibility and aperiodicity of the corresponding Markov chain; Conditions (G2), (A1), and (A2) ensure reversibility and Condition (A3) ensures that stationary distributions concentrate on the set of optimal solutions as \(c\) approaches 0. In general terms, the conditions require the following. Condition (G1) requires that each solution can be reached from any other solution by generating a finite sequence of neighbouring solutions. Clearly, to ensure this, the corresponding neighbourhood graph should be strongly connected. Condition (G2) requires symmetry of the generation matrix. Conditions (A1)-(A3) require that the acceptance matrix is well-behaved, i.e., improvements are always accepted and deteriorations are accepted...
with finite probability for \( c > 0 \) (Condition (A1)) and with zero probability for \( \lim c \downarrow 0 \) (Condition (A3)); the factorization required by Condition (A2) ensures detailed balance as defined by (8).

The Conditions (G1)-(A3) are sufficient but not necessary. Thus, there may be acceptance and generation matrices not satisfying these conditions and still ensuring the existence of the stationary distribution. An example of such an acceptance matrix is

\[
A_{ij}(c) = \frac{1}{1 + \exp(-(f(i) - f(j))/c)}. \tag{13}
\]

This acceptance matrix does not satisfy Conditions (A1) and (A2), but it can be shown to lead to the stationary distribution of (9) by using the detailed balance equations of (8).

Furthermore, several authors have addressed the generality issue of the acceptance probability. We discuss this issue following the lines of the work by Schuur [1989].

**Theorem 3.4** Let \( P(c) \) be the transition matrix of the homogeneous Markov chain associated with the simulated annealing algorithm defined by (4) and let Conditions (G1) and (G2) of Theorem 3.3 hold. Furthermore, let the acceptance probabilities be defined as follows.

There exist functions \( \phi : (0, \infty) \times \mathbb{R} \rightarrow (0, \infty) \) and \( H : (0, \infty) \times \mathbb{R} \times \mathbb{R} \rightarrow (0, 1] \), such that for \( c > 0 \), and \( x, y \in \mathbb{R} : H(c, x, y) = H(c, y, x) \), and

\[
A_{ij}(c) = H(c, f(i), f(j)) \min \left(1, \frac{\phi(c, f(j))}{\phi(c, f(i))} \right),
\]

and

\[
\forall x, y \in \mathbb{R} : x > y \Rightarrow \lim_{c \downarrow 0} \frac{\phi(c, y)}{\phi(c, x)} = 0. \tag{14}
\]

Then the Markov chain has a unique stationary distribution \( q(c) \), whose components are given by

\[
q_i(c) = \frac{\phi(c, f(i))}{\sum_{j \in S} \phi(c, f(j))} \quad \text{for all } i \in S, \tag{15}
\]

and

\[
\lim_{c \downarrow 0} q_i(c) = q_i^* \]

where the \( q_i^* \) are again given by (10).

As a corollary of Theorem 3.4 it is argued that the only well-behaved function \( \phi(c, f(j)) \) that satisfies (14) is of the form

\[
\phi(c, f(j)) = \exp(\gamma(c)f(j)), \tag{16}
\]

where \( \gamma : (0, \infty) \rightarrow (0, \infty) \) and \( \lim_{c \downarrow 0} \gamma(c) = \infty \).

Several authors have argued that the fastest convergence to the stationary distribution of (15) with (16) and \( \gamma = c^{-1} \) is given by the acceptance probabilities of (6) [Kesidis & Wong, 1990; Romeo & Sangiovanni-Vincentelli, 1991].

30
Below we give alternatives for Conditions (G1) and (G2), respectively.

Condition (G1) can be replaced by a more general condition. As already mentioned, this condition ensures that the Markov chain associated with the generation matrix $G$ is irreducible. If this is not the case, asymptotic convergence to a subset of the set of globally optimal solutions can still be proved if Condition (G1) is replaced by the following necessary and sufficient condition.

$$(G1') \quad \forall i \in S, \exists i^* \in S^*, p \geq 1, \exists l_0, l_1, \ldots, l_p \in S,$$

with $l_0 = i, l_p = i^*$, and

$$G_{l_k l_{k+1}} > 0, \quad k = 0, 1, \ldots, p - 1.$$

According to this condition it should be possible to construct a finite sequence of transitions with non-zero generation probability, leading from an arbitrary solution $i$ to some optimal solution $i^*$. For the proof of the validity of this condition, a distinction must be made between transient and recurrent solutions, where a solution is called transient if the probability that the Markov chain ever returns to that solution equals zero, and recurrent if the Markov chain may return to the solution with a finite probability [Feller, 1950]. Furthermore, the stationary distribution of (12) does not apply any more and should be replaced by a stationary matrix $Q(c)$ whose elements $q_{ij}$ denote the probability of finding a solution $j$ after an infinite number of transitions, starting from a solution $i$. A more detailed treatment of this is considered to be beyond the scope of this chapter. For a detailed treatment of this subject the reader is referred to [Connors & Kumar, 1987; Gidas, 1985; Van Laarhoven, 1988; Van Laarhoven, Aarts & Lenstra, 1992].

In practice one does not want to bother with the requirement of Condition (G2) that the generation matrix must be symmetric. More easy to implement is a uniform distribution over the neighbourhoods, similar to that used in the original version of simulated annealing. For that case Lundy & Mees [1986] show that for the following choice of the generation probabilities

$$G_{ij} = \frac{1}{|\mathcal{N}(i)|} \chi_{\mathcal{N}(i)}(j), \quad \text{for all } i, j \in S,$$  \hspace{1cm} (17)

Condition (G2) is no longer needed to ensure asymptotic convergence, and that the components of the stationary distribution are then given by

$$q_i(c) = \frac{|\mathcal{N}(i)|}{\sum_{j \in S} |\mathcal{N}(j)| A_{ij}(c)/A_{jj}(c)}.$$

Moreover, it follows directly that these components again converge to the $q_i^*$ of (10) as $c \downarrow 0$. Finally, we mention that a generation matrix satisfying both Condition (G2) and (17) implies that $|\mathcal{N}(i)|$ is independent of $i$.

### 3.4 The inhomogeneous model

In the previous section it was shown that, under certain conditions on the generation and acceptance matrices, the simulated annealing algorithm converges to a global minimum with
probability 1, if for each value of the control parameter \( c_k, k = 0,1,2,\ldots \), the corresponding homogeneous Markov chain is infinitely long and the sequence \( (c_k | k = 0,1,2,\ldots) \) eventually converges to 0 as \( k \to \infty \). In this section we discuss conditions to ensure asymptotic convergence for the case where each Markov chain is of length 1, i.e., after each transition the value of the control parameter is changed. Thus, the simulated annealing algorithm is modelled as an inhomogeneous Markov chain with the following transition probabilities.

\[
P_{ij}(c_k) = \begin{cases} 
G_{ij}(c_k)A_{ij}(c_k) & j \neq i \\
1 - \sum_{l \notin \{i\}} G_{il}(c_k)A_{il}(c_k) & j = i.
\end{cases}
\] (18)

Furthermore, we assume that the sequence \( (c_k | k = 0,1,2,\ldots) \) satisfies the following two conditions.

\[
\lim_{k \to \infty} c_k = 0, \quad c_k \geq c_{k+1}, \quad k = 0,1,\ldots
\]

Thus, we do not exclude the possibility that \( c_k \) is kept constant during a number of transitions, in which case we again obtain a homogeneous Markov chain, but of finite length.

We show that, under certain conditions on the rate of convergence of the sequence \( (c_k | k = 0,1,2,\ldots) \), the inhomogeneous Markov chain associated with the simulated annealing algorithm converges in distribution to \( q^* \), whose components are given by (10). In other words we prove that

\[
\lim_{k \to \infty} \mathbb{P}\{X(k) \in S^*\} = 1.
\]

To discuss the convergence of inhomogeneous Markov chains we need the following definitions; see Seneta [1981].

**Definition 3.8** Let \( P(k) \) be the transition matrix associated with an inhomogeneous Markov chain on a set of outcomes \( \mathcal{O} \). Then the matrix \( U(m,k) \) is defined as

\[
U(m,k) = \prod_{n=m}^{k} P(n), \quad 0 < m \leq k.
\]

In other words the components of \( U(m,k) \) are equal to

\[
U_{ij}(m,k) = \mathbb{P}\{X(k) = j | X(m-1) = i\}, \quad \text{for all} \quad i,j \in \mathcal{O}
\]

**Definition 3.9** (ergodicity) A finite inhomogeneous Markov chain on a set of outcomes \( \mathcal{O} \) is weakly ergodic if

\[
\forall i,j,l \in \mathcal{O}, \forall m > 0: \lim_{k \to \infty} (U_{il}(m,k) - U_{jl}(m,k)) = 0.
\]

It is strongly ergodic if there exists a stochastic vector \( q^* \), such that

\[
\forall i,j \in \mathcal{O}, \forall m > 0: \lim_{k \to \infty} U_{ij}(m,k) = q_j^*.
\]
Thus, for a given $m$, weak ergodicity implies that $X(k)$ becomes independent of $X(m)$ as $k \to \infty$, whereas strong ergodicity implies convergence in distribution, i.e., for any stochastic vector $a(m)$ denoting the probabilities of the outcomes of the $m$th trial we have that

$$\lim_{k \to \infty} a^T(m - 1) \prod_{n=m}^k P(n) = (q^*)^T,$$

or

$$\lim_{k \to \infty} P\{X(k) = j\} = \lim_{k \to \infty} \left(\sum_{i \in \mathcal{O}} U_{ij}(m, k)P\{X(m - 1) = i\}\right) = q_j^*, \text{ for all } ij \in \mathcal{O}.$$

The difference between weak and strong ergodicity can be understood from the following example. Let the transition probabilities $P_{ij}(k)$ of an inhomogeneous Markov chain be independent of $j$. Then the Markov chain is clearly weakly ergodic but it is not strongly ergodic if the $P_{ij}(k)$ vary for ever with $k$ for a given $i$. Note that for a homogeneous Markov chain there is no distinction between weak and strong ergodicity.

The following two theorems provide conditions for weak and strong ergodicity of inhomogeneous Markov chains. The proofs can be found in [Seneta, 1981] or [Isaacson & Madsen, 1976].

**Theorem 3.5** Let $r_1(X)$ denote the coefficient of ergodicity of the $n \times n$ stochastic matrix $X$ defined as

$$r_1(X) = \frac{1}{2} \max_{i,j=1,\ldots,n} \sum_{l=1}^n |X_{ii} - X_{jl}|$$

$$= 1 - \min_{i,j=1,\ldots,n} \sum_{l=1}^n \min(X_{ii}, X_{jl}).$$

Then an inhomogeneous Markov chain is weakly ergodic if and only if there is a strictly increasing sequence of positive numbers $(k_i \mid i = 0, 1, 2, \ldots)$, such that

$$\sum_{i=0}^{\infty} (1 - r_1(X(k_i, k_{i+1}))) = \infty,$$

(21)

**Theorem 3.6** A finite inhomogeneous Markov chain is strongly ergodic under the following conditions.

(C1) the Markov chain is weakly ergodic,

(C2) for all $k$ there exists a stochastic vector $q(k)$ such that $q(k)$ is the left eigenvector of $P(k)$ with eigenvalue 1, and

(C3) the eigenvectors $q(k)$ satisfy

$$\sum_{k=1}^{\infty} \|q(k) - q(k + 1)\|_1 < \infty,$$

where the 1-norm of an $n$-vector $x$ is defined as $\|x\|_1 = \sum_{i=1}^n |x_i|$. 

33
Moreover, if \( q^* = \lim_{k \to \infty} q(k) \), then \( q^* \) is the vector in Definition 3.9.

To prove convergence in distribution for the simulated annealing algorithm it must be shown that the associated inhomogeneous Markov chain is strongly ergodic.

**Theorem 3.7** Let \((S, f)\) be an instance of a combinatorial optimization problem, \(N\) a neighbourhood structure, and \(P(k)\) the transition matrix of the inhomogeneous Markov chain associated with the simulated annealing algorithm defined by (18), (5), and (6). Furthermore, let the following conditions be satisfied.

\[
(D1) \quad \forall i, j \in S, \exists p \geq 1, \exists l_0, l_1, \ldots, l_p \in S, \\
\text{with } l_0 = i, l_p = j, \text{ and} \\
G_{l_k, l_{k+1}} > 0, \quad k = 0, 1, \ldots, p - 1.
\]

\[
(D2) \quad c_k \geq \frac{\Gamma}{\log(k+k_0)}, \quad k = 0, 1, \ldots, \tag{23}
\]

for some value of \( \Gamma > 0 \) and \( k_0 > 2 \).

Then the Markov chain converges in distribution to the vector \( q^* \), with components given by (10), or, in other words

\[
\lim_{k \to \infty} \mathbb{P}\{X(k) \in S^*\} = 1. \tag{24}
\]

Theorem 3.7 can be proven by showing that Conditions (D1) and (D2) are sufficient to satisfy Conditions (C1), (C2), and (C3) of Theorem 3.5, according to the following argumentation.

Condition (D1) guarantees the existence of the left eigenvector \( q(k) \) of \( P(k) \), given by \( q(k) = q(c_k) \), i.e., the stationary distribution of the homogeneous Markov chain with transition matrix \( P = P(k) \); see Theorem 3.2. The components of the eigenvectors are given by (9). Furthermore, from (19) and (9) we have \( \lim_{k \to \infty} q(c_k) = \lim_{c \to 0} q(c) = q^* \), where the components of \( q^* \) are given by (10). So, Condition (C2) of Theorem 3.6 holds and by using the explicit form of the eigenvectors \( q(k) \), Condition (C3) can be shown to hold. What remains to be shown is that the Markov chain is weakly ergodic; Condition (C1). This can be done by using Theorem 3.5 and Condition (D2).

The proof of this is quite technical and several authors have come up with different approaches, which vary predominantly in the estimations of the values of the constant \( \Gamma \).

One of the first results was obtained by Mitra, Romeo & Sangiovanni-Vincentelli [1986]. To discuss this we need the following definition

**Definition 3.10** The distance \( d(i, j) \) between two solutions \( i, j \in S \) is defined as the length \( d \) of the shortest sequence of solutions \((l_0, l_1, \ldots, l_d)\), with \( l_0 = i, l_d = j, \) and \( P_{l_m, l_{m+1}}(k) > 0, l_m \in S, m = 0, 1, \ldots, d - 1. \)
Mitra, Romeo & Sangiovanni-Vincentelli [1986] found that $\Gamma \geq r\Delta$ with

$$\Delta = \max_{i \in S} \max_{j \in \mathcal{N}(i)} \{|f(j) - f(i)|\},$$

and

$$r = \min_{i \in S\setminus S^*} \max_{j \in S} d(i, j),$$

where $\mathcal{S}$ denotes the set of all locally minimal solutions.

Other values of $\Gamma$ are given by Anily & Federgruen [1987a], Gelfand & Mitter [1985], Geman & Geman [1984], Gidas [1985], and Holley & Stroock [1988]; for an overview see [Romeo & Sangiovanni-Vincentelli, 1991].

The conditions for asymptotic convergence given above are sufficient but not necessary. Necessary and sufficient conditions are derived by Hajek [1988]. The essential difference with the sufficient conditions again lies in the difference in value of the constant $\Gamma$. To discuss Hajek's result we need the following definitions.

**Definition 3.11** Let $i, j \in S$, then $j$ is reachable at height $h$ from $i$ if $i = j$ and $f(i) \leq h$ or $\exists p \geq 1$, $\exists l_0, \ldots, l_p \in S$ with $l_0 = i$ and $l_p = j$, such that $G_{l_k, l_{k+1}} > 0$ and $f(l_k) \leq h$, for all $k = 0, \ldots, p - 1$.

**Definition 3.12** Let $i$ be a local minimum, then the depth $d(i)$ of $i$ is the smallest number $x, x > 0$, such that there is a solution $j \in S$ with $f(j) < f(i) + x$ from $i$. By definition $d(i^*) = \infty$.

We now can formulate the results obtained by Hajek.

**Theorem 3.8** Let $(c_k | k = 0, 1, \ldots )$ be a sequence of values of the control parameter defined as

$$c_k = \frac{\Gamma}{\log(k + 2)}, \quad k = 0, 1, \ldots,$$

for some constant $\Gamma$. Then asymptotic convergence of the simulated annealing algorithm, using the transition probabilities of (18), (5) and (6), is guaranteed if and only if

- the Markov chain is irreducible,

- $i$ is reachable from $j$ at height $h$ if and only if $j$ is reachable from $i$ at height $h$, for arbitrary $i, j \in S$ and $h$, and

- the constant $\Gamma$ satisfies $\Gamma \geq D$, where

$$D = \max_{i \in \tilde{S} \setminus S^*} d(i),$$

i.e., $D$ is the depth of the deepest local, non-global minimum.

Kern [1986] has addressed the problem of calculating the value of $D$. In particular, he showed for a number of problems that it is unlikely that $D$ can be calculated in polynomial time for

35
arbitrary instances of a combinatorial optimization problem. In addition, the paper presents bounds on the value of $D$ for several combinatorial optimization problems.

Finally, we mention that, under certain conditions, asymptotic convergence of the inhomogeneous Markov chain associated with the simulated annealing algorithm can also be proved for general conditions on the generation and acceptance probabilities. This result was first proved by Anily & Federgruen [1987b] and can be formulated as follows.

**Theorem 3.9** Let the transition probabilities of the inhomogeneous Markov chain associated with the simulated annealing algorithm be defined by (18) and let the generation probabilities $G_{ij}(c)$ and acceptance probabilities $A_{ij}(c)$ satisfy Conditions (G1) - (A3) of Theorem 3.3. Furthermore, let

$$
\Delta(c) = \min_{i,j \in S} \{ A_{ij}(c) | i \in S, j \in N(i) \}.
$$

Then

$$
\lim_{k \to \infty} \mathbb{P}\{X(k) \in S^*\} = 1,
$$

if

$$
\sum_{k=0}^{\infty} (\Delta(c_k))^n = \infty,
$$

where $n$ denotes the maximum number of steps needed to reach an optimal solution from any arbitrary solution, and the constant $\Gamma$ assumes the following form.

$$
\Gamma \leq n\Delta,
$$

where $\Delta$ is given by (25).

Finally we discuss a result due to Gelfand & Mitter [1985], who derived sufficient conditions for convergence to an arbitrary set of solutions $\mathcal{I}$. To discuss their results we need the following definitions.

**Definition 3.13** Let $(S, f)$ be an instance of a combinatorial optimization problem, $N$ a neighbourhood structure and $P(k)$ the transition matrix of the inhomogeneous Markov chain associated with the simulated annealing algorithm defined by (18). Furthermore, let $I \subset S$ be an arbitrary set of solutions. Then for any pair of solutions $i, j \in S$, $T^{(d)}_{ij}$ is defined as the set of all sequences $r$ of length $d$ such that $r = (l_0, l_1, \ldots, l_d)$, $l_m \in S$, $m = 0, 1, \ldots, d - 1$, with $l_0 = i, l_d = j$, and $P_{l_m l_{m+1}}(k) > 0$; see also Definition 3.10. Furthermore, for $i, j \in S$, $d \in \mathbb{Z}^+$ and $r \in T^{(d)}_{ij}$ let

$$
\Gamma(r) = \sum_{m=0}^{d-1} \max(0, f(l_{m+1}) - f(l_m)),
$$

and

$$
\Gamma^{(d)}_{ij} = \begin{cases} 
\min_{r \in T^{(d)}_{ij}} \Gamma(r) & \text{if } T^{(d)}_{ij} \neq \emptyset \\
\infty & \text{otherwise},
\end{cases}
$$

and finally

$$
\Gamma_{ij} = \min_{d \in \mathbb{Z}^+} \Gamma^{(d)}_{ij}.
$$
Let \( J \) be the set of solutions, not belonging to \( I \), i.e., \( J = S \setminus I \). If a lowercase index of \( \Gamma \) is replaced by a set, then an additional minimization is to be carried out over the elements of the corresponding set, e.g.,

\[
\Gamma^{(d)}_{ij} = \min_{j \in J} \Gamma^{(d)}_{ij}.
\]

The results obtained by Gelfand & Mitter can now be formulated as follows.

**Theorem 3.10** Let the transition probabilities of the inhomogeneous Markov chain associated with the simulated annealing algorithm be defined by (18) and the acceptance probabilities \( A_{ij}(c) \) be defined by (6). Furthermore, let the generation probabilities be such that the following conditions hold.

- \( \exists d \in \mathbb{Z}^+, \forall j \in J : \Gamma^{(d)}_{jj} = \Gamma_{jj} \),
- \( \max_{j \in J} \Gamma_{jj} < \infty \),
- \( \forall j \in J : \Gamma_{jj} < \Gamma_{IJ} \), and
- \( \varepsilon_k \leq \frac{1}{\log(k+k_0)} \), where \( \Gamma = \max_{j \in J} \Gamma_{jj} \) and \( k_0 > 2 \).

Then we have

\[
\lim_{k \to \infty} \mathbb{P}\{X(k) \in I\} = 1.
\]

Note that the rate of convergence of the sequence \( (\varepsilon_k)_{k=0,1,\ldots} \) in Theorem 3.10 is again inverse logarithmic as in the sequences in Theorems 3.7 and 3.8.

### 3.5 Asymptotic behaviour

In the previous sections we have shown that the simulated annealing algorithm converges in probability to the set of optimal solutions, or in other words, asymptotically the algorithm finds an optimal solution with probability one. As a result of the limits of (11) or (24), asymptotic convergence to the set of optimal solutions is achieved only after an infinite number of transitions. In any finite time implementation this is clearly impossible and one must therefore resort to approximations of the asymptotic convergence.

With respect to the approximation of the stationary distribution we have the following two properties [Seneta, 1981].

**Property 3.1** Let \( P(c) \) denote the transition matrix of the homogeneous Markov chain associated with the simulated annealing algorithm defined by (4) and \( q(c) \) denote the corresponding stationary distribution given by the left eigenvector with eigenvalue \( 1 \) of \( P(c) \). Then, as \( k \to \infty \), we have

\[
\|a(k) - q(c)\|_1 = O(k^s|\lambda_2(c)|^k), \tag{30}
\]

where \( a(k) \) denotes the probability distribution of the outcomes after \( k \) trials, \( \lambda_2(c) \) \( (0 < |\lambda_2(c)| < 1) \) denotes the second largest eigenvalue of \( P(c) \) with multiplicity \( m_2 \), and \( s = m_2 - 1 \).

Hence, the speed of convergence to the stationary distribution is determined by \( \lambda_2(c) \). Unfortunately, computation of \( \lambda_2(c) \) is impracticable, due to large size of the matrix \( P(c) \). Approximation of the norm in (30) leads to the following property [Aarts & Van Laarhoven, 1985].

37
Property 3.2 Let $\varepsilon$ denote an arbitrarily small positive number, then

$$\|a(k) - q(\varepsilon)\|_1 < \varepsilon,$$

if

$$k > K \left( 1 + \frac{\ln\left(\frac{1}{\varepsilon} \frac{1}{\ln(1 - \gamma(\varepsilon))}\right)}{\ln(1 - \gamma(\varepsilon))} \right),$$

where $\gamma(\varepsilon) = \min_{i,j \in S} P^+_{ij}(\varepsilon)$ and $K = |S|^2 - 3|S| + 3$.

Hence, (31) and (32) indicate that the stationary distribution is approximated arbitrarily closely, only if the number of transitions is at least quadratic in the size of the solution space. Moreover, this size $|S|$ is for most problems exponential in the size of the problem itself, for instance in TRAVELLING SALESMAN $|S| = (n - 1)!$, where $n$ denotes the number of cities. Thus, the analysis presented above indicates that approximating the stationary distribution arbitrarily closely results in an exponential-time execution of the simulated annealing algorithm.

With respect to the asymptotic convergence of the inhomogeneous Markov chain associated with the simulated annealing algorithm we have the following result [Mitra, Romeo & Sangiovanni-Vincentelli, 1986].

Property 3.3 Let the transition probabilities of the inhomogeneous Markov chain associated with the simulated annealing algorithm be defined by (18), (5) and (6), and let the sequence $(c_k | k = 0, 1, \ldots)$ be given by (23), with $\Gamma > r\Delta$, where $r$ and $\Delta$ are defined as in (25) and (26). Furthermore, let $q^*$ be the uniform probability distribution on the set of optimal solutions defined by (10). Then for $k \to \infty$,

$$\|a(k) - q^*\|_1 < \varepsilon,$$

for an arbitrarily small positive number $\varepsilon$, if

$$k = O \left( \varepsilon^{-\max(a,b)} \right),$$

where

$$a = \frac{r^{r\Delta/\Gamma}}{w^r}, \quad and \quad b = \frac{r\Delta}{\hat{f} - f^*},$$

with $\hat{f} = \min_{i \in S \setminus S^*} f(i)$, and $w = \min_{i \in S} \min_{j \in N(i)} G_{ij}$.

Evaluation of this bound for particular problem instances typically leads to a number of transitions that is larger than the size of the solution space and thus to an exponential-time execution for most problems. For instance, in the case of the TRAVELLING SALESMAN we have [Aarts & Korst, 1989a]

$$k = O \left( n^{2n-1} \right).$$

Note that $|S| = O(n^n)$. Hence, complete enumeration of all solutions would take less time than approximating an optimal solution arbitrarily closely by the simulated annealing algorithm.

Summarizing, we have shown that the simulated annealing algorithm behaves as an optimization algorithm if it is allowed an infinite number of transitions. Approximating the asymptotic
behaviour arbitrarily closely, requires a number of transitions that for most problems is typically larger than the size of the solution space, leading to an exponential-time execution of the algorithm. Thus, the simulated annealing algorithm is clearly unsuited for solving combinatorial optimization problems to optimality. However, in the next chapter we will show how the asymptotic behaviour of the simulated annealing algorithm can be approximated in polynomial time. Obviously, this is at the cost of the guarantee of obtaining optimal solutions, but, as will be shown, the simulated annealing algorithm using this polynomial-time approximation returns high-quality solutions for most problem instances.

3.6 Finite-time approximation

In the previous section we have shown that implementation of the simulated annealing algorithm as an optimization algorithm requires an infinite number of transitions. We now focus on finite-time implementations, thus resulting in an approximation algorithm.

A finite-time implementation of the simulated annealing algorithm is obtained by generating a sequence of homogeneous Markov chains of finite length at descending values of the control parameter. For this, a set of parameters must be specified that govern the convergence of the algorithm. These parameters are combined in a so-called cooling schedule.

Definition 3.14 A cooling schedule specifies a finite sequence of values of the control parameter, and a finite number of transitions at each value of the control parameter, i.e., more precisely

- an initial value of the control parameter $c_0$,
- a decrement function for decreasing the value of the control parameter,
- a final value of the control parameter specified by a stop criterion, and
- a finite length of each homogeneous Markov chain.

Central to the discussion of cooling schedules is the concept of quasi equilibrium. Let $L_k$ denote the length of the $k^{th}$ Markov chain and $c_k$ the corresponding value of the control parameter. Then quasi equilibrium is achieved if the probability distribution $a(L_k, c_k)$ of the solutions, after $L_k$ trials of the $k^{th}$ Markov chain, is 'sufficiently close' to the stationary distribution at $c_k$, i.e.,

$$||a(L_k, c_k) - q(c_k)||_1 < \varepsilon,$$

for some specified positive value of $\varepsilon$.

From Property 3.2 we recall that a number of transitions is required that is quadratic in the size of the solution space in order to satisfy (33) for arbitrarily small values of $\varepsilon$, which leads to an exponential-time execution for most problems. Thus, in order to be of practical use, a less rigid quantification of the quasi equilibrium concept is needed than that of (33). For this one may resort to the following arguments. For the acceptance probabilities of (6), and well-behaved
generation probabilities, the stationary distribution is of the form given by (9). For \( c \to \infty \), the stationary distribution is given by a uniform distribution on the set of solutions \( S \), i.e.,

\[
\lim_{c \to \infty} q_c(c) = \frac{|N(i)|}{\sum_{j \in S} |N(j)|}.
\]

Thus, at sufficiently large values of \( c_k \) - allowing acceptance of virtually all proposed transitions - quasi equilibrium is obtained by definition, since all solutions occur with equal probability given by the uniform distribution of (34). Next, the decrement function and the Markov chain lengths must be chosen such that quasi equilibrium is restored at the end of each individual Markov chain. In this way the equilibrium distributions for the various Markov chains will be 'closely followed', so as to arrive eventually, as \( c_k \downarrow 0 \), close to \( q^* \), the uniform distribution on the set of optimal solutions given by (10).

It is intuitively clear that large decrements in \( c_k \) will require longer Markov chain lengths in order to restore quasi equilibrium at the next value of the control parameter, \( c_{k+1} \). Thus, there is a trade-off between large decrements of the control parameter and small Markov chain lengths. Usually, one goes for small decrements in \( c_k \) to avoid extremely long chains, but alternatively, one could use large values for \( L_k \) in order to be able to make large decrements in \( c_k \).

The search for adequate cooling schedules has been the subject of many studies over the past years. For reviews the reader is referred to [Van Laarhoven & Aarts, 1987; Collins, Eglese & Golden, 1988; Romeo & Sangiovanni-Vincentelli, 1991].

The schedules presented in the literature can be divided into two broad classes: static and dynamic schedules. In a static cooling schedule the parameters are fixed; they cannot be changed during execution of the algorithm. In a dynamic cooling schedule the parameters are adaptively changed during execution of the algorithm. Below we present some examples.

**Static cooling schedules**

**Initial Value of the Control Parameter.** To ensure a sufficiently large value of \( c_0 \), one may choose \( c_0 = \Delta f_{\text{max}} \), where \( \Delta f_{\text{max}} \) is the maximal difference in cost between any two neighbouring solutions. Exact calculation of \( \Delta f_{\text{max}} \) is quite time consuming in many cases. However, one often can give simple estimates of its value.

**Decrement of the control parameter.** A frequently used decrement function is given by

\[
c_{k+1} = \alpha \cdot c_k, \quad k = 0, 1, \ldots,
\]

where \( \alpha \) is a positive constant smaller than but close to 1. Typical values lie between 0.8 and 0.99.

**Final value of the control parameter.** The final value is fixed at some small value, which may be related to the smallest possible difference in cost between two neighbouring solutions.

**Markov chain length.** The length of Markov chains is fixed by some number that may be related to the size of the neighbourhoods in the problem at hand.

The schedule given above is clearly very simple. The geometric decrement rule originates from
the schedule proposed in the first paper by Kirkpatrick, Gelatt & Vecchi [1983] and is still used in many practical situations.

**Dynamic cooling schedules**

One may think of a number of basic extensions of the above schedule that lead to a dynamic schedule. For instance, a sufficiently large value of $c_0$ may be obtained by requiring that the initial acceptance ratio $\chi_0$ - defined as the number of accepted at $c_0$ - is close to 1. This can be achieved by starting off at a small positive value of $c_0$ and multiplying it with a constant factor, larger than 1, until the corresponding value of $\chi_0$, that is calculated from a number of generated transitions, is close to 1. Typical values of $\chi_0$ lie between 0.9 and 0.99. An adaptive calculation of the final value of the control parameter may be obtained by terminating the execution of the algorithm at a $c_k$-value for which the value of the cost function of the solution obtained in the last trial of a Markov chain remains unchanged for a number of consecutive chains. Clearly such a value exists for each local minimum that is found. The length of Markov chains may be determined by requiring that at each value $c_k$ a minimum number of transitions is accepted. However, since transitions are accepted with decreasing probability, one would obtain $L_k \to \infty$ for $c_k \to 0$. Therefore, $L_k$ is usually bounded by some constant $L_{\text{max}}$ to avoid extremely long Markov chains for small values of $c_k$.

In addition to this basic dynamic schedule the literature presents a number of more elaborate schedules. Most of these schedules are based on a statistical analysis of the simulated annealing process, thus allowing a more theoretical estimation of the parameters. For the transition probabilities of (4), (5), and (6), the statistical analysis leads to a model for the cost distribution that resembles an exponential distribution at low $c$-values and a normal distribution at high $c$-values. Within this model the first two moments of the resulting distribution are given by [Aarts, Korst & Van Laarhoven, 1988].

\[
\mathbb{E}_c(f) = \mathbb{E}_\infty(f) - \frac{\sigma^2_\infty(f)}{c} \left( \frac{\gamma c}{\gamma c + 1} \right),
\]

and

\[
\sigma^2_c(f) = \sigma^2_\infty(f) \left( \frac{\gamma c}{\gamma c + 1} \right)^2,
\]

where $\gamma$ is given by

\[
\gamma = \frac{\mathbb{E}_\infty(f) - f^*}{\sigma^2_\infty(f)}.
\]

Note that $\mathbb{E}_\infty(f)$ and $\sigma^2_\infty(f)$ can be simply replaced by the average cost value and the corresponding standard deviation, respectively.

With the above analysis a number of adaptive parameter estimates have been derived. As an example we discuss the schedule proposed by Huang, Romeo & Sangiovanni-Vincentelli [1986], since this schedule is quoted in the literature as the most efficient one among those whose implementation require only a modicum of sophistication. For instance, the schedule by Lam & Delosme [1986] is conjectured to be even more efficient but its intricacy generally hinders practical use.
Initial Value of the Control Parameter. From (35) it follows directly that $\mathbb{E}_c(f) \approx \mathbb{E}_\infty(f)$ for $c \gg \sigma^2_\infty(f)$. Hence $c_0$ may be chosen as

$$c_0 = K \sigma^2_\infty(f),$$

where $K$ is a constant typically ranging from 5 to 10.

Decrement of the control parameter. Here the concept of quasi equilibrium is quantified by requiring that the average cost difference for two consecutive Markov chains is small, i.e., $\mathbb{E}_{c_{k+1}}(f) - \mathbb{E}_{c_k}(f) = -\epsilon$ for some small positive number $\epsilon$. Next, by using

$$\frac{\partial}{\partial \ln c} \mathbb{E}_c(f) = \frac{\sigma^2_c(f)}{c},$$

and replacing the left hand side of (36) by the differential quotient, we obtain

$$\frac{\mathbb{E}_{c_{k+1}}(f) - \mathbb{E}_{c_k}(f)}{\ln c_{k+1} - \ln c_k} = \frac{\sigma^2_{c_k}(f)}{c_k}.$$

This results a decrement rule given by

$$c_{k+1} = c_k \exp \left( - \frac{\epsilon c_k}{\sigma^2_{c_k}(f)} \right),$$

where, for practical purposes, $\sigma_{c_k}(f)$ is approximated by the measured deviation. In their original paper, Huang, Romeo & Sangiovanni-Vincentelli [1986] replace $\epsilon$ by $\lambda \sigma_{c_k}$, $\lambda < 1$, which gives only a slight modification of (37).

Final value of the control parameter. Execution is terminated if at the end of a Markov chain

$$f'_{\text{max}} - f'_{\text{min}} = \Delta f'_{\text{max}},$$

where $f'_{\text{max}}$ and $f'_{\text{min}}$ denote the maximum and minimum cost value, respectively, and $\Delta f'_{\text{max}}$ the maximum cost difference of the solutions accepted during the generation of that chain. If (38) holds, $c$ is set to 0 and the execution is concluded with a simple local search to ensure local optimality of the final solution.

Markov chain length. Statistical analysis leads to the observation that in equilibrium the fraction of solutions generated with cost values within a certain range $\epsilon$ from the expected cost reaches a stationary value $\kappa$. Assuming a normal distribution of the cost values, Huang, Romeo & Sangiovanni-Vincentelli [1986] show that $\kappa = -\text{erf}(\epsilon/\sigma_c(f))$, where erf$(x)$ is the error function [Abramowitz & Stegun, 1971]. The Markov chain length is determined by the number of trials $L_k$ for which

$$L_k^* = p \kappa,$$

where $p$ is a parameter depending on the size of the problem instance, and $L_k^*$ is defined as the number of accepted solutions with a cost value within the interval $(\mathbb{E}_c - \epsilon, \mathbb{E}_c + \epsilon)$. An additional bound on $L_k$ is introduced to avoid extremely long Markov chains.
3.7 Simulated annealing in practice

To apply simulated annealing in practice four basic ingredients are needed: a concise problem representation, a neighbourhood structure, a transition mechanism, and a cooling schedule. The algorithm is usually implemented as a sequence of homogeneous Markov chains of finite length, generated at descending values of the control parameter specified by the cooling schedule. No general rules are known that guide the choice of these items and it is evident that the skill of an annealing practitioner lies in the way they are handled.

In its ten years of existence, simulated annealing has been widely applied to a large variety of problems in such diverse areas as planning, design, engineering, biology and physics. So far about a thousand papers have been published reporting on practical applications of the algorithm. Many of these studies have led to modifications of the algorithm such as the use of penalty functions, alternative generation and acceptance probabilities, implementation specific aspects, parallel versions, etc. Due to the large variety of approaches and the many different (implementation) details, it is impractical to give a balanced in-depth overview of the practical experience that has been gathered. We therefore restrict ourselves here to some general statements and a listing of some appropriate references.

To start with the latter, we refer for overviews on applications of simulated annealing to Aarts & Korst [1989a], Collins, Eglese & Golden [1988], Van Laarhoven & Aarts [1987]. Eglese [1990] presents a general overview of the practical experience with the algorithm based on computational results. Other studies emphasizing computational results can be found in Van Laarhoven [1988]. Johnson, Aragon, McGeoch & Schevon [1989; 1991] report on an extensive numerical study for several combinatorial optimization problems including GRAPH PARTITIONING, GRAPH COLOURING and NUMBER PARTITIONING. Perhaps the most striking element in this work is the observed performance ambivalence. For GRAPH PARTITIONING simulated annealing seems to outperform all existing approximation algorithms, whereas for NUMBER PARTITIONING the performance is hopelessly poor. Although this bad performance for NUMBER PARTITIONING can be understood from analytical arguments, there seems no way to adapt the algorithm in order to improve on it. A similar ambivalence is encountered in the area of code design, where for FOOTBALL POOL simulated annealing is able to improve on the best known results [Van Laarhoven, Aarts, Van Lint & Wille, 1988]. On the other hand, for problems related to the design of binary sequences, the algorithm is drastically inferior to simple constructive methods [Beenker, Claassen & Hermens, 1985]. Finally, we refer to the first paper of Johnson, Aragon, McGeoch & Schevon [1989] for a list of practical findings in the use of simulated annealing that in our opinion reflects the general experience of annealing practitioners.

Broadly speaking, simulated annealing is for many applications capable of finding good solutions but often at the cost of substantial running times. The merits of the algorithm has become especially pronounced in the many successes in industrial problem settings, where time is of no concern. As an example we mention design problems, since for these problems one is primarily interested in finding high-quality solutions, whereas design time often only plays a minor role. A well-known example of the use of simulated annealing in such areas is that of VLSI design [Sechen & Sangiovanni Vincentelli, 1985; Wong, Leong & Liu, 1988]. The success of simulated annealing can be explained from the fact that the algorithm is easy to implement and capable of handling
almost any optimization problem and any constraint, either by appropriate neighbourhoods or by relaxation through the use of penalty functions. These properties are however not unique for simulated annealing. They also hold for simple local search algorithms. The main advantage of simulated annealing is that it is able to improve upon the relatively poor performance of local search by simply replacing the deterministic (strict improvement) acceptance criterion by a stochastic criterion, thus circumventing the need of an in-depth study of the problem structure in order to construct more effective neighbourhoods, or to design more tailored algorithms. It goes without saying that in an industrial environment this is a great advantage, since often the required expertise is not available and even more importantly, it saves time.

3.8 Related topics

The literature presents many variations on the basic simulated annealing approach presented in the previous sections. Many of these variations concentrate on alternatives that should reduce the potentially burdensome amount of time required by simulated annealing algorithm to converge to near-optimal solutions. Roughly speaking the existing approaches fit into three categories: fast sequential algorithms, hardware acceleration, and parallel algorithms. Below, we mention a few examples.

Szu & Hartley [1987] present an annealing algorithm for the optimization of continuous-valued functions, using a generation mechanism given by a Cauchy distribution instead of the usually used normal distribution. They claim that their generation mechanism leads to an inverse linear cooling rate instead of an inverse logarithmical cooling rate as was found for the Gaussian distribution. This approach has been further refined by Ingber [1989], who proposes a simulated re-annealing technique that permits an exponential cooling rate.

Another example of an improved generation mechanism is the rejectionless method introduced by Green & Supowit [1986]. These authors propose to generate new solutions with a probability proportional to the effect of a transition on the cost function. In this way, a subsequent solution is directly chosen from the neighbourhood of a given solution; no rejection of solutions will occur. This method leads to shorter Markov chains in a number of problems. However, the efficient use of the method depends strongly on some additional conditions on the neighbourhood structure, which, unfortunately, cannot be met by many combinatorial optimization problems.

Spira & Hage [1985], propose to speed up the execution of simulated annealing by implementing time-consuming parts of the algorithm in micro code to be executed on a fast general purpose micro engine attached to a host computer. Other hardware accelerations are obtained by VLSI implementations of simulated annealing along the lines of neural networks; see for instance Lee & Sheu [1991]. This subject is readdressed in the next section.

Parallel simulated annealing algorithms aim at distributing the execution of the various parts of a simulated annealing algorithm over a number of communicating parallel processors. This is a promising approach to the problem of speeding up the execution of the algorithm, but it is by no means a trivial task, due to the intrinsic sequential nature of the algorithm, i.e., transitions must be carried out one after the other. Over the years a large variety of approaches have been proposed, leading to both generally applicable and tailored algorithms. For overviews we refer
to Aarts & Korst [1989a], Azencott [1992], and Greening [1990]. We return to this issue in Section 4.

Finally, we mention two variations that are based on a combination of simulated annealing with the multi-start paradigm. Martin, Otto & Felten [1991] propose a simulated annealing algorithm for the TRAVELLING SALESMAN using a restricted 4-exchange neighbourhood, combined with a simple local search algorithm using a 3-exchange neighbourhood. Eiben, Aarts & Van Hee [1991] present a stochastic search procedure that combines elements of population genetics with those of simulated annealing. This stochastic approach to genetic algorithms exhibits similar convergence properties to simulated annealing.
4 Boltzmann Machines

One of the main conclusions reached for simulated annealing is that the algorithm can obtain high-quality solutions but often at the cost of large running times. As the need to solve large problem instances inevitably will increase, the situation with respect to computational effort will only worsen. To cope with this situation, research has been concentrating on the investigation of possibilities for speeding up simulated annealing by execution on parallel machines. This has led to a variety of parallel annealing algorithms either following the concept of data parallelism or that of function parallelism.

In data parallelism, parallelism is achieved by using several parallel processes that carry out identical functions on distributed data structures. The use of this concept in local search requires the definition of distributed neighbourhood structures. Examples of such structures are known for TRAVELLING SALESMAN [Verhoeven, Aarts, Van der Sluis & Vaessens, 1992] and GRAPH PARTITIONING [Savage & Wloka, 1991].

In function parallelism, parallelism is achieved by using several parallel processes that carry out different functions on a common distributed structure. In the context of local search, this concept of parallelism has been applied almost exclusively to simulated annealing. Overviews can be found in [Aarts & Korst, 1989a; Azencott, 1992; Greening, 1990; Kravitz & Rutenbar, 1987; Roussel-Ragot & Dreyfus, 1990]. So far the successes obtained with parallel annealing algorithms are moderate. Most parallel algorithms presented in the literature follow rather strictly the basic concept of the annealing algorithm, i.e., generation of a sequence of solutions, which is intrinsically sequential and thus leads to rather low efficiencies on parallel machines. Our general feeling is that the design of efficient parallel annealing algorithms requires a computational model that differs from the traditional sequential annealing model.

A promising approach in this respect is provided by neural network models, which are based on an analogy with information processing in the human brain. These models follow the assumption that information can be processed by massively parallel networks consisting of co-operating simple neuron-like computing elements that are highly interconnected by weighted links. Another typical feature is that the response of an individual computing element to other elements in the network is given by a non-linear scalar function of the weighted activity of the elements it is connected to. Over the years the work on neural networks has led to many different models. Overviews can be found in the books by Rumelhart & McClelland [1986], and Grossberg [1988].

One of the emergent neural network models is the Boltzmann machine, introduced by Hinton & Sejnowski [1983]. It is a symmetric neural network model that uses a randomized state transition mechanism, which shows a strong similarity with the acceptance criterion in a simulated annealing algorithm. This, together with the fact that the computing elements of a Boltzmann machine can operate simultaneously, makes the model of Boltzmann machines a sensible candidate for the computational model required above.

Interest in Boltzmann machines extends over a number of disciplines including combinatorial optimization, classification, and learning [Ackley, Hinton & Sejnowski, 1985; Hinton & Sejnowski,
1986]. For a more extensive treatment of each of these items the reader is referred to our book [Aarts & Korst, 1989a]. In this chapter we confine ourselves to a discussion on the relation between Boltzmann machines and combinatorial optimization.

4.1 Hopfield networks

In the same way as simulated annealing is a randomized version of local search, Boltzmann machines are randomized versions of a class of deterministic neural networks known as Hopfield networks [Hopfield, 1982]. In general terms, a Hopfield network consists of a set of computing elements that are connected in some way. The computing elements are logic units with two discrete states, viz. "on" or "off". The units are linked by symmetric connections. With each connection a strength is associated representing a quantitative measure for the hypothesis that the two units interconnected by the link are both "on". A consensus function assigns to a configuration of the network - determined by the states of the individual units - a real number which is a quantitative measure for the amount of consensus in the network with respect to the set of underlying local hypotheses. In the original paper the consensus was called energy [Hopfield, 1982] and this notion is still widely used by others. The units may change their states in order to maximize the consensus. A state change of an individual unit is determined by a deterministic response function of the states of its adjacent units, i.e., the ones it is connected to, and the associated connection strengths.

Structural description

A Hopfield network consists of a number of two-state units, that are connected in some way. The network can be represented by a pseudograph \( \mathcal{H} = (\mathcal{U}, \mathcal{C}) \), where \( \mathcal{U} \) denotes a finite set of units and \( \mathcal{C} \subseteq \mathcal{U} \times \mathcal{U} \) a set of symmetric connections between units. A connection \( \{u, v\} \in \mathcal{C} \) joins the units \( u \) and \( v \). The set of connections may contain loops or bias connections, i.e., \( \{u, u\} \mid u \in \mathcal{U} \} \subset \mathcal{C} \). If two units are connected, they are called adjacent. A unit \( u \) can be in two states, either in state "0" or state "1", corresponding to "off" and "on", respectively.

Definition 4.1 A configuration \( k \) of a Hopfield network is a global state which is uniquely defined by a sequence of length \( |\mathcal{U}| \), whose \( u \)th component \( k(u) \) denotes the state of unit \( u \) in configuration \( k \). The configuration space \( \mathcal{R} \) is given by the set of all possible configurations. Clearly, the cardinality of \( \mathcal{R} \) equals \( 2^{|\mathcal{U}|} \).

Definition 4.2 Let \( \{u, v\} \in \mathcal{C} \) be a connection joining units \( u \) and \( v \), then \( \{u, v\} \) is activated in a given configuration \( k \) if both \( u \) and \( v \) are "on", i.e., if

\[ k(u) \cdot k(v) = 1. \]

Definition 4.3 With each connection \( \{u, v\} \in \mathcal{C} \) a connection strength \( s_{\{u, v\}} \in \mathbb{R} \) is associated as a quantitative measure for the desirability that \( \{u, v\} \) is activated. By definition, \( s_{\{u, v\}} = s_{\{v, u\}} \). If \( s_{\{u, v\}} > 0 \), it is desirable that \( \{u, v\} \) is activated; if \( s_{\{u, v\}} < 0 \), it is undesirable. Connections with a positive and negative strength are called excitatory and inhibitory, respectively. Furthermore, the strength \( s_{\{u, u\}} \) of a bias connection \( \{u, u\} \) is called the bias of unit \( u \).
Definition 4.4 (consensus) The consensus function $C: \mathbb{R} \rightarrow \mathbb{R}$ assigns to each configuration $k$ a real number, called the consensus, which equals the sum of the strengths of the activated connections, i.e.,

$$C(k) = \sum_{(u,v) \in \mathcal{C}} s_{(u,v)} k(u) k(v).$$

(39)

The optimal consensus is denoted by $C^*$. The set of optimal configurations is denoted by $R^*$. For reasons of notation, we denote $C(k)$ by $C_k$.

In general the consensus will be large if many excitatory connections are activated, and it will be small if many inhibitory connections are activated. In fact, the consensus is a global measure indicating to what extent the units in the network have reached a consensus about their individual states, subject to the desirabilities expressed by the individual connection strengths. Since the connection strengths impose local constraints, these networks are also often called constraint satisfaction networks [Hinton & Sejnowski, 1983]. The basic idea of implementing local constraints as connection strengths in networks dates back to the work of Moussouris [1974].

The aim of the network is to reach a globally optimal configuration, i.e., a configuration with maximal consensus. To reach such a configuration, a state transition mechanism is used which allows units to change their states, from "0" to "1" or vice versa, in order to adjust their states to those of the adjacent units. Let the network be in configuration $k$, then changing the state of unit $u$ results in a configuration $l$, with $l(u) = 1 - k(u)$ and $l(v) = k(v)$ for $v \neq u$. Furthermore, let $C_u$ denote the set of connections incident with unit $u$, excluding $(u,u)$. Then the difference in consensus induced by changing the state of unit $u$ in configuration $k$, is given by

$$\Delta C_k(u) = r_k(u) h_k(u),$$

(40)

where

$$r_k(u) = 1 - 2k(u),$$

(41)

and

$$h_k(u) = \sum_{(u,v) \in C_u} s_{(u,v)} k(v) + s_{(u,u)}. $$

(42)

From (40)-(42) it is clear that the effect on the consensus, resulting from changing the state of unit $u$, is entirely determined by the states of its adjacent units and the corresponding connection strengths. Consequently, each unit can locally evaluate its state transition since no global calculations are required.

In a Hopfield network the response of an individual unit $u$ to its adjacent units in a given configuration $k$ is typically given by a deterministic function $B_k(u)$ of the following form.

$$B_k(u) = \begin{cases} 1 & \text{if } \Delta C_k(u) \geq 0 \\ 0 & \text{if } \Delta C_k(u) < 0 \end{cases}.$$  

(43)

The relationship between Hopfield networks and iterative improvement in local search is apparent; see Sections 2 and 3. The solution space is given by the $2^{[u]}$ configurations of the network. The "cost" of each configuration is given by its consensus, which must be maximized. For
each configuration a neighbourhood is given by the set of configurations that can be reached by changing the state of one of the units in that configuration. The objective is to maximize the consensus by subsequent state transitions, accepting only those transitions that lead to an improvement in consensus.

**Definition 4.5** A locally maximal configuration, or simply local maximum, is defined as a configuration $k \in \mathcal{R}$ with

$$\Delta C_k(u) \leq 0 \quad \text{for all } u \in \mathcal{U}.$$  

The set of all local maxima is denoted by $\mathcal{R}$.

**Complexity issues**

The problem of finding a stable configuration in a Hopfield network can be formulated as a PLS-problem in the following way; see Section 2.4 and Definitions 2.10 and 2.12.

**Definition 4.6** (STABLE CONFIGURATION) Given a Hopfield network $\mathcal{H} = (\mathcal{U}, \mathcal{C})$ with connection strengths $s_{\{u,v\}}$ for each $\{u,v\} \in \mathcal{C}$, and let

- $\mathcal{R} = \{k \mid k \in \{0,1\}^{2|\mathcal{U}|}\}$,
- $C_k = \sum_{\{u,v\} \in \mathcal{C}} s_{\{u,v\}} k(u) k(v)$, for all $k \in \mathcal{R}$,
- $\mathcal{N}(k) = \{l \in \mathcal{R} \mid d(k,l) \leq 1\}$, for all $k \in \mathcal{R}$,
  where $d(k,l)$ gives the number of units that have different states in $k$ and $l$,
- $\forall k \in \mathcal{R}$:
  - $F_a(k) = (1, \ldots, 1)$,
  - $F_b(k) = C_k$, and
  - $F_c(k) = \begin{cases} k & \text{if } C_k = \min_{m \in \mathcal{N}(k)} C_m \\ l & \text{otherwise,} \end{cases}$
  where $l \in \mathcal{N}(k)$ and $C_l = \min_{m \in \mathcal{N}(l)} C_m$.

Find a local minimum of $C$.

Schäffer & Yannakakis [1991] give a slightly different formulation of STABLE CONFIGURATION. However, the differences are only minor. Schäffer & Yannakakis [1991] have shown that STABLE CONFIGURATION is PLS-complete by a PLS-reduction from MAXCUT. Since STABLE CONFIGURATION is equivalent to the problem of finding a local maximum for the Hopfield network, it follows directly that finding a local maximum for the Hopfield network may require exponential running times in the worst case. A special subclass of STABLE CONFIGURATION is given by those instances for which all the edge weights equal $-1$. Schäffer & Yannakakis [1991] have shown that this subclass is P-complete.

Goles-Chacc, Fogelman-Soulie & Pellegrin [1985] have analysed the speed of convergence of Hopfield networks. They found that the number of sequential iterations $L$ needed to find a locally maximal configuration of the network is bounded by

$$L \leq \sum_{\{u,v\} \in \mathcal{C}} |s_{\{u,v\}}| + 2 \sum_{\{u,u\} \in \mathcal{C}} |s_{\{u,u\}}|. \tag{44}$$
Clearly, this results in pseudo-polynomial running times, which implies that there are networks with connection strengths for which $L$ is exponential in the size of the network, hence leading to an exponential number of iterations. Goles-Chacc, Fogelman-Soulie & Pellegrin [1985] also considered the fully parallel case in which within one iteration all units may adjust their states. For this situation no stable configuration may exist, but the network may converge to a cycle of length at most two. The number of iterations is of the same order as (44). Bruck & Goodman [1988] consider a generalized convergence theorem that unifies the two cases mentioned above.

Boltzmann machine are randomized versions of Hopfield networks. Parberry & Schnitger [1989] have shown that \textit{clocked} Boltzmann machines - in the next section we will call this synchronously parallel Boltzmann machines - can be simulated by a standard unbounded fan-in threshold circuit with polynomially bounded size and running time greater by a constant factor. Parberry & Schnitger obtain their result in the following three steps. First, they show that cycles can be removed from the connection network of a Boltzmann machine. Next, they show how randomness can be removed by a standard sampling technique. Finally, the connection strengths are removed resulting in a standard combinational circuit. This result implies that Boltzmann machines are not much more powerful than standard threshold circuits. It shows that the use of randomization and computationally more complex units are relatively unimportant features with respect to computational power of the entire network. They only lead to a gain by a constant factor in speed and a polynomially bounded factor in size. However, it should be taken into account that threshold circuits constitute an extremely powerful parallel model of computation compared to the more classical parallel models [Reif, 1987].

### 4.2 Boltzmann machines

A Boltzmann machine has a network structure identical to that of a Hopfield network. The main difference with Hopfield networks lies in the response function, which, in the case of a Boltzmann machine is given by a probability function. For the Hopfield network this is a deterministic function; c.f. (43).

**Definition 4.7** The response in a Boltzmann machine of an individual unit $u$ to its adjacent units in a configuration $k$ is given by

$$A_k(u, c) = P_c\{\text{accept a state change of unit } u \mid k\}$$

$$= \frac{1}{1 + \exp(-\Delta C_k(u)/c)},$$

(45)

where $\Delta C_k(u)$ is given by (40) and $c \in \mathbb{R}^+$ denotes a control parameter, playing a role similar to the control parameter of simulated annealing; see also Section 3.

As in simulated annealing, the probability function of (45) allows a Boltzmann machine to escape from locally optimal configurations, so as to arrive eventually at a globally optimal configuration. The response function is slightly different from the classical acceptance probability used in simulated annealing; see Definition 3.4. One of the reasons of this is that it more closely
resembles the sigmoid response of biological neurons and that it is easier to implement in VLSI hardware.

Again the theory of Markov chains can be used to describe the state transitions of the units in a Boltzmann machine leading asymptotically to a configuration with maximal consensus. To this end we distinguish between the following two models.

- **Sequential Boltzmann machines**, in which units may change their states only one at a time, and

- **Parallel Boltzmann machines**, in which units may change their states simultaneously.

In the following two sections we elaborate on these two models in more detail. The discussion of the convergence of Boltzmann machines is based on the homogeneous simulated annealing model presented in Section 3.

### 4.3 Sequential Boltzmann Machines

In a sequential Boltzmann machine, units may change their states only one at a time. Hence, given a configuration $k$, then a *neighbouring configuration* $ku$ is obtained by changing the state of a unit $u \in \mathcal{U}$, i.e.,

$$ku(v) = \begin{cases} k(v) & \text{if } v \neq u \\ 1 - k(v) & \text{if } v = u. \end{cases}$$

A trial in a sequential Boltzmann machine is an attempt of a single unit to change its state. The corresponding Markov transition probabilities then can be defined as follows.

**Definition 4.8** Let $k$ and $l$ be two configurations of a sequential Boltzmann machine, then the transition probability $P_{kl}(c)$ of transforming $k$ into $l$ is defined as follows; see also (4).

$$P_{kl}(c) = \Pr\{X(m) = l \mid X(m - 1) = k\},$$

$$P_{kl}(c) = \begin{cases} G(u)A_k(u, c) & \text{if } l = ku \\ 1 - \sum_{u \in \mathcal{U}} G(u)A_k(u, c) & \text{if } l = k \\ 0 & \text{otherwise}, \end{cases}$$

(46)

where $G(u)$ denotes the generation probability, i.e., the probability of generating a state transition of unit $u$, and $A_k(u, c)$ is the acceptance probability, i.e. the probability of accepting the proposed state transition of unit $u$ in configuration $k$.

The acceptance probability $A_k(u, c)$ is chosen according to (45). The generation probability is usually chosen uniformly over the available units and independently of the configuration $k$ and the control parameter $c$. This choice is motivated primarily by the fact that we would like the generation probability to be as simple as possible. All units in a Boltzmann machine are considered to be identical and consequently there is no reason beforehand to treat some units differently from others with respect to generating state transitions. This concept is formalized by the notion fairness, which is introduced in Definition 4.16.
We now can prove that the sequential Boltzmann machine converges asymptotically to the set of globally optimal configurations.

**Theorem 4.1** Let the transition probabilities in a Boltzmann machine be given by (46), with uniform generation probabilities and acceptance probabilities given by (45). Then

- there exists a unique stationary distribution $q(c)$ for all $c > 0$, whose components are given by
  \[ q_k(c) = \frac{\exp(C_k/c)}{\sum_{i \in k} \exp(C_i/c)}, \]  
  \[ (47) \]

and

- the stationary distribution converges as $c \downarrow 0$ to a uniform distribution over the set of optimal configurations, i.e.,
  \[ \lim_{c \downarrow 0} q_k(c) \overset{\text{def}}{=} q_k^* = \frac{1}{|R^*|} \chi(R^*)(k), \]  
  \[ (48) \]

where $R^*$ denotes the set of optimal configurations, or in other words, the Boltzmann machine finds with probability 1 a configuration with maximal consensus.

The proof of Theorem 4.1 is identical to that of Theorem 3.2. From the choice of the generation probabilities it follows directly that each configuration can be generated from any other configuration in a finite number of steps. Hence, the associated Markov chain is irreducible; see Definition 3.6. Furthermore, using (7), (45), and (46), it follows that the Markov chain is also aperiodic. Consequently, the conditions of Theorem 3.1 are satisfied, resulting in the existence of a unique equilibrium distribution. Next, the detailed balance equations of (8) can be used to complete the first part of the theorem. The proof of the second part follows directly from (47).

The result presented in Theorem 4.1 relates to the asymptotic case, thus requiring an infinite number of transitions. As in the simulated annealing algorithm, a finite-time approximation is obtained by specifying a set of parameters that determine a cooling schedule; see [Aarts & Korst, 1989a]. As in simulated annealing, such a finite-time approximation no longer guarantees convergence to an optimal configuration.

### 4.4 Parallel Boltzmann Machines

To model parallelism in a Boltzmann machine one may distinguish between synchronous and asynchronous state transitions.

- **Synchronous state transitions**: sets of state transitions are scheduled in successive trials, each trial consisting of a number of individual state transitions. After each trial the accepted state transitions are communicated through the network so that all units have up-to-date information about the states of their neighbours before the next trial is initiated. During each trial a unit is allowed to propose a state transition exactly once. Evidently, synchronous parallelism requires a global clocking scheme to control the synchronization.
Asynchronous parallelism: state transitions are evaluated simultaneously and independently. Units continuously generate state transitions and accept or reject them on the basis of information that is not necessarily up-to-date, since the states of its neighbours may have changed in the meantime. Clearly, asynchronous parallelism does not require a global clocking scheme, which is of advantage in hardware implementations.

Below we restrict the discussion to synchronous state transitions. Asynchronous transitions cannot be modelled by Markov chains and require a completely different approach, which is considered beyond the scope of this chapter. An extensive discussion of synchronously parallel Boltzmann machines is given by Zwietering & Aarts [1991]. Here, we only briefly summarize the most important results.

Definition 4.9 Let for each pair of configurations \( k, l \in R \), \( U_{kl} \) be the subset of units that must change their states in order to transform \( k \) into \( l \), i.e.,

\[
U_{kl} = \{ u \in U | k(u) \neq l(u) \}.
\]

Then the transition probabilities of a synchronously parallel Boltzmann machine are given by

\[
P_{kl}(c) = \sum_{U \supseteq U_{kl}} G(U_s) A_{kl}(U_s, c),
\]

where \( G(U_s) \) denotes the generation probability for the set \( U_s \subseteq U \), and \( A_{kl}(U_s, c) \) the acceptance probability for the units in \( U_s \); \( c \) again denotes the control parameter mentioned above.

The choices for the generation and acceptance probabilities play an important role in the modelling of parallel Boltzmann machines, and the discussion in the remainder of this section is centered on these two items.

The generation probability

The generation probabilities determine the amount of parallelism in a Boltzmann machine and the way the units are synchronized. Furthermore, the outcome of the generation process, i.e., the set \( U_s \), defines for each configuration \( k \) a neighbourhood \( \mathcal{R}_k(U_s) \) consisting of all configurations \( l \in R \) that can be reached from \( k \) by a state transition of one or more units in \( U_s \), i.e.,

\[
\mathcal{R}_k(U_s) = \{ m \in R | U_{km} \subseteq U_s \}.
\]

Furthermore, we impose the following conditions on \( G \).

\[
\forall U_s \subseteq U : G(U_s) \geq 0, \quad (50)
\]

\[
\sum_{U_s \subseteq U} G(U_s) = 1, \quad (51)
\]

\[
\bigcup_{G(U_s) > 0} U_s = U. \quad (52)
\]

The first two conditions ensure that \( G \) is a correctly defined probability function. The last condition guarantees that each unit can be in the set of units that is allowed to make a transition.
The acceptance probability

As a direct consequence of the probability function of the individual units given in (45), the acceptance probabilities assume the following form.

\[ \forall k, l \in \mathcal{R}, \forall \mathcal{U}_s \supseteq \mathcal{U}_{kl} : A_{kl}(\mathcal{U}_s, c) = \prod_{u \in \mathcal{U}_{kl}} A_k(u, c) \prod_{u \in \mathcal{U} \setminus \mathcal{U}_{kl}} (1 - A_k(u, c)). \] (53)

Zwietering & Aarts [1991] show that (53) can be rewritten as

\[ \forall \mathcal{U}_s \supseteq \mathcal{U}_{kl} : A_{kl}(\mathcal{U}_s, c) = \frac{\exp(C_l/c) \exp(E_{kl}/c)}{\sum_{m \in \mathcal{R}_k(\mathcal{U}_s)} \exp(C_m/c) \exp(E_{km}/c)}, \] (54)

where

\[ E_{kl} = - \sum_{\{u,v\} \in \mathcal{E}, u \neq v} s_{\{u,v\}}(k(u) - l(u))(k(v) - l(v)). \]

From (54) it follows directly that the matrix \( A(\mathcal{U}_s, c) \) is stochastic. Together with (51), this implies that the transition matrix \( P(c) \) given by (49) is stochastic.

Furthermore, it can be easily verified that the conditions of (50)-(52) together with the definition of the acceptance probabilities of (45) ensure that the Markov chain induced by (49) is irreducible and aperiodic, and thus by Theorem 3.1, there exists a unique stationary distribution.

Finally, the expressions of (54) reveal the following two important aspects.

- If \( E_{km} = 0 \) for all \( m \in \mathcal{R}_k(\mathcal{U}_s) \), then the acceptance probabilities are similar to those of a sequential Boltzmann machine and thus the same convergence properties will hold, i.e., asymptotic convergence to a configuration with optimal consensus.

- If \( E_{km} \neq 0 \) for some \( m \in \mathcal{R}_k(\mathcal{U}_s) \), then it is not (yet) clear what the behaviour of the synchronously parallel Boltzmann machine will be.

Roughly speaking, \( E_{kl} \) can be viewed as an error term in the transition probability resulting from "incorrect" calculation of the consensus difference between configurations \( k \) and \( l \), which is due to the simultaneous state transitions of adjacent units. Below we elaborate on these aspects in more detail and partly answer the question raised by the second item.

So far we know that the Markov chain that can be associated with a synchronously parallel Boltzmann machine with transition probabilities given by (49) and generation and acceptance probabilities defined by (50)-(52), and (49), respectively, has a unique stationary distribution which we denote by \( q(c) \). In general it is desirable to have an analytical expression for \( q(c) \), in order to derive its asymptotic behaviour. Unfortunately, such an expression could not be obtained for the general case. Using a different approach the outcome of \( \lim_{c \to 0} q(c) \) for the general case can be conjectured. Furthermore, the conjecture is true in those cases for which an analytical expression of \( q(c) \) can be obtained. Before we can formulate these results we need the following definitions.
Definition 4.10 For each configuration \( k \in \mathcal{R} \) and subset \( \mathcal{U}_s \subseteq \mathcal{U} \) the configuration \( k_{\mathcal{U}_s}^* \in \mathcal{R}_k(\mathcal{U}_s) \) is defined as

\[
k_{\mathcal{U}_s}^*(u) = \begin{cases} 
1 & \text{if } h_k(u) > 0 \text{ and } u \in \mathcal{U}_s \\
0 & \text{if } h_k(u) < 0 \text{ and } u \in \mathcal{U}_s \\
k(u) & \text{if } h_k(u) = 0 \text{ or } u \notin \mathcal{U}_s,
\end{cases}
\] (55)

where \( h_k \) is given by (42).

Definition 4.11 (pseudo consensus) For every configuration \( k \in \mathcal{R} \) and subset \( \mathcal{U}_s \subseteq \mathcal{U} \), the pseudo consensus function \( S_k(\mathcal{U}_s) \) is defined as

\[
S_k(\mathcal{U}_s) = D_{kk_{\mathcal{U}_s}^*},
\] (56)

where

\[
D_{kl} = 2 \sum_{\{u,v\} \in E} s_{\{u,v\}} k(u)l(u) + \sum_{u \in \mathcal{U}} s_{\{u,u\}} |k(u) - l(u)|.
\]

In fact, \( k_{\mathcal{U}_s}^* \) in Definition 4.11 is the configuration \( l \in \mathcal{R}_k(\mathcal{U}_s) \) for which \( D_{kl} \) is maximal. Furthermore, (55) implies that \( k_{\emptyset}^* = k \) and hence from (56) it follows that \( S_k(\emptyset) = D_{kk} = 2C_k \).

Definition 4.12 (extended consensus) For each configuration \( k \in \mathcal{R} \) the extended consensus function \( \tilde{C}_k \) is defined as

\[
\tilde{C}_k = \sum_{\mathcal{U}_s \subseteq \mathcal{U}} G(\mathcal{U}_s)S_k(\mathcal{U}_s).
\] (57)

The set of configurations \( k \in \mathcal{R} \) for which the extended consensus function is maximal is denoted by \( \mathcal{R}^* \).

We now come to our main result, which consists in proofs of the following conjecture for various interesting special cases.

Conjecture 4.1 Let \( B \) be a synchronously parallel Boltzmann machine with transition probabilities given by (49), and generation and acceptance probabilities defined by (50)-(52), and (45), respectively. Then the probability distribution of configurations visited in the associated Markov chain converges as \( c \downarrow 0 \) to a distribution over the set \( \mathcal{R}_1 \), or in other words the Boltzmann machine finds with probability 1 a configuration in \( \mathcal{R}_1 \).

A proof of this conjecture is an open problem. However, for two special subclasses of synchronously parallel Boltzmann machines, correctness can be proved. For this we distinguish between the following two cases.

- **Limited parallelism**: units may change their states simultaneously only if they are not adjacent.

- **Unlimited parallelism**: units may change their states simultaneously whether or not they are adjacent.

**Limited parallelism**

For this case the stationary distribution \( q(c) \) can be expressed analytically, which allows the verification of the conjecture.

55
Definition 4.13 A synchronous parallel Boltzmann machine is said to apply limited parallelism if the generation probabilities satisfy the following condition.

\[ \forall U_s \subseteq U : G(U_s) > 0 \Rightarrow \forall u, v \in U_s, u \neq v : \{u, v\} \notin \mathcal{C}. \]  (58)

This definition yields the following result [Aarts & Korst, 1989a; Zwietering & Aarts, 1991].

Theorem 4.2 Let B be a synchronously parallel Boltzmann machine applying limited parallelism. Then

\[ \forall U_s \subseteq U : G(U_s) > 0 \Rightarrow \forall k \in \mathcal{R}, l \in \mathcal{R}_k(U_s) : E_{kl} = 0. \]  (59)

This leads to the following theorem.

Theorem 4.3 Let B be a synchronously parallel Boltzmann machine applying limited parallelism with transition probabilities given by (49), and generation and acceptance probabilities defined by (50)-(52), and (45), respectively. Then

- there exists a stationary distribution of the finite homogeneous Markov chain induced by P(c) on \( \mathcal{R} \), whose components are given by

\[ \forall k \in \mathcal{R} : q_k(c) = \frac{\exp(C_k/c)}{\sum_{m \in \mathcal{R}} \exp(C_m/c)}, \]

and

- the stationary distribution converges as \( c \downarrow 0 \) to a (uniform) distribution over the set of optimal configurations, i.e., \( \lim_{c \downarrow 0} q(c) = q^* \), where the components of \( q^* \) are given by (48).

The proof of this theorem is similar to that of Theorem 4.1. Furthermore, with some calculus it can be shown that in this case \( \hat{\mathcal{R}}^* = \mathcal{R}^* \), which implies that Conjecture 4.1 is true for the case of limited parallelism.

Unlimited parallelism

For a synchronous Boltzmann machine with unlimited parallelism, we cannot explicitly calculate the stationary distribution as we did for Boltzmann machines with limited parallelism; see previous section. Therefore, we follow a different approach.

To this end, we consider the special case in which the matrix \( A(U_s, c) \) induces a finite homogeneous Markov chain on the subspace \( \mathcal{R}_{k_0}(U_s) \) of \( \mathcal{R} \), where \( k_0 \in \mathcal{R} \) is an arbitrary but fixed configuration. This is a valid Markov process since \( \mathcal{R}_{k_0}(U_s) \) is a recurrent subspace and \( A(U_s, c) \) is a stochastic matrix. This process corresponds to a Boltzmann machine in which at each trial the same "isolated" subset of units \( U_s \) is generated. This special case is then generalized to the general case of unlimited parallelism.

Definition 4.14 The functions \( J_{kk}, g_k(U_s, c) \) and \( Z_k(U_s) \), for all \( k \in \mathcal{R}, U_s \subseteq \mathcal{U} \) and \( c > 0 \), are defined as

\[ J_{kk}(U_s) = 2C_k + \frac{1}{2} \sum_{u \in U_s} r_k(u) h_k, \]
Theorem 4.4 Let $k_0 \in R$ be a configuration and $\mathcal{U}_s \subseteq \mathcal{U}$ a subset of units. Consider the subspace $R_{k_0}(\mathcal{U}_s)$ and transition matrix $P = A(\mathcal{U}_s, c)$. Then

- there exists a unique stationary distribution $q(\mathcal{U}_s, c)$ of the finite homogeneous Markov chain induced by $P$ on $R_{k_0}(\mathcal{U}_s)$, whose components are given by

$$ q_k(\mathcal{U}_s, c) = \begin{cases} \frac{\exp(g_k(\mathcal{U}_s, c))}{\sum_{m \in R_{k_0}(\mathcal{U}_s)} \exp(g_m(\mathcal{U}_s, c))} & \text{if } k \in R_{k_0}(\mathcal{U}_s) \\ 0 & \text{if } k \notin R_{k_0}(\mathcal{U}_s) \end{cases} $$

and

- the stationary distribution converges as $c \downarrow 0$ to a distribution over the set of configurations $k \in R_{k_0}(\mathcal{U}_s)$ for which the pseudo consensus function $S_k(\mathcal{U}_s)$ is maximal, i.e., $\lim_{c \downarrow 0} q(\mathcal{U}_s, c) = q^*(\mathcal{U}_s)$, where

$$ q_k^*(\mathcal{U}_s) = \begin{cases} \frac{Z_k(\mathcal{U}_s)}{\sum_{m \in R_{k_0}(\mathcal{U}_s)} Z_m(\mathcal{U}_s)} & \text{if } k \in R_{k_0}^*(\mathcal{U}_s) \\ 0 & \text{if } k \notin R_{k_0}^*(\mathcal{U}_s) \end{cases} $$

where $R_{k_0}^*(\mathcal{U}_s)$ denotes the set of configurations in the subclass $R_{k_0}(\mathcal{U}_s)$ with maximal extended consensus, i.e., $R_{k_0}^*(\mathcal{U}_s) = \{ k \in R_{k_0}(\mathcal{U}_s) : \forall l \in R_{k_0}(\mathcal{U}_s) : S_l(\mathcal{U}_s) \leq S_k(\mathcal{U}_s) \}$.

The proof of this theorem is again similar to that of Theorem 4.1.

We can use Theorem 4.4 to provide some evidence for the correctness of Conjecture 4.1. Instead of viewing $P(c)$ defined by (49) as a single matrix that induces a single Markov chain on the configuration space $R$, one might consider it as a collection of matrices $A(\mathcal{U}_s, c)$, each of which induces a Markov chain on its corresponding subspace $R_k(\mathcal{U}_s)$. In this perspective, we view $G$ as a selecting mechanism between the different Markov chains; the Markov chain induced by $A(\mathcal{U}_s, c)$ is selected with probability $G(\mathcal{U}_s)$.

Theorem 4.4 states that the selected Markov chain induced by $A(\mathcal{U}_s, c)$ maximizes $S_l(\mathcal{U}_s)$ over the subspace $R_k(\mathcal{U}_s)$. Due to the fact that the generation may yield a different subset each time it is applied, where, because of conditions (50)-(52), each unit has a positive probability of being in that subset, the maximization becomes effective over the whole configuration space. Hence,
$P(c)$ corresponds to a process where the function $S_k(U_s)$ is maximized with probability $G(U_s)$ as $c \downarrow 0$. In other words, $P(c)$ corresponds to the process of maximizing $\bar{C}_k$ as $c \downarrow 0$.

As already mentioned, a proof of Conjecture 4.1 is still an open problem. However, in the special case of full unlimited parallelism, which we define below, the correctness follows directly from Theorem 4.4.

**Definition 4.15** A synchronous parallel Boltzmann machine is said to apply full parallelism if the generation probability satisfies $G(U_s) = 1 \leftrightarrow U_s = U$.

**Theorem 4.5** Conjecture 4.1 is true for a synchronously parallel Boltzmann machine with full parallelism.

The proof follows as a corollary of Theorem 4.4, since in the case of full parallelism we have $P(c) = A(U,c)$ and $\bar{C}_k = S_k(U)$. Hence, using $R_{k_0}(U) = R$ for all $k_0 \in R$, it follows directly that Conjecture 4.1 and the second part of Theorem 4.5 are identical.

Conjecture 4.1 has some interesting implications for a large class of synchronously parallel Boltzmann machines. For this we consider the concept of fairness.

**Definition 4.16 (fairness)** A synchronously parallel Boltzmann machine is called fair if all units have equal probabilities of proposing a state transition, in other words there exists a constant $\alpha \in (0,1]$ such that

$$\forall u \in U : \sum_{u_s \subseteq U} G(U_s)X_{u_s}(u) = \alpha. \quad (60)$$

Fairness is a natural condition for the generation mechanism of a Boltzmann machine. In fact it can be straightforwardly implemented by randomly generating the assembly of the sets $U_s$. Furthermore, in a fair synchronously parallel Boltzmann machine, the constant $\alpha$ of (60) equals the expected fraction of units that is allowed to make a transition in each trial. Zwietering & Aarts [1991] have proven the following theorem.

**Theorem 4.6** Let $B$ be a fair synchronously parallel Boltzmann machine. Then

- the extended consensus function $\bar{C}$ defined by (57) satisfies for all $k \in R$

$$\bar{C}_k = (1 - \alpha)2C_k + \alpha S_k,$$

where $S_k = S_k(U)$ and $\alpha$ is the value defined in (60), and

- if $\alpha_c$ is defined by

$$\alpha_c = \min_{k \in R \setminus R^*} \frac{2C^* - 2C_k}{S_k - 2C_k}, \quad S_k \geq 2C^*$$

then $\bar{R}^* = R^*$ if and only if $\alpha < \alpha_c$.

So the theorem states that the extended consensus function of a fair Boltzmann machine can be written as a weighted sum of the consensus function of (39) and the pseudo consensus function $S_k = S_k(U)$ of (56). The critical value $\alpha_c$ provides an upper bound for the maximum amount
of parallelism that can be safely exploited in a synchronously parallel Boltzmann machine for maximization of the consensus function of \((39)\). This is an important issue for the use of Boltzmann machines in combinatorial optimization; see Section 4.5. Essential here is the relation between the cost function of the optimization problem at hand and the consensus function of \((39)\). More precisely, it is shown that the relation is based on a mapping of an instance of the optimization problem onto a Boltzmann machine such that maximization of the consensus is equivalent to optimization of the cost function. So it is essential that parallel state transitions indeed lead to a maximization of the consensus.

Zwietering & Aarts [1991] have derived the following simple bounds for the value of \(\alpha_c\).

\[
\frac{2C^* - 2C^\dagger}{S^* - 2C^\dagger} \leq \alpha_c \leq \frac{2C^*}{S_0},
\]

where \(S_0 = \mathcal{S}_0(U) = \sum_{u \in U} s^\dagger_{\{u, u\}}\) denotes the pseudo consensus of the configuration with all units "off", \(S^* = \max_{k \in \mathcal{R}} S_k\), and \(C^\dagger = \max_{k \in \mathcal{R} \setminus \mathcal{C}} C_k\). In the next section we give an example of the estimation of \(\alpha_c\) for INDEPENDENT SET.

### 4.5 Combinatorial optimization and Boltzmann machines

To map an instance of a combinatorial optimization problem onto a Boltzmann machines, we need to choose an appropriate network structure and corresponding connection strengths, such that the problem of finding (near) optimal solutions of the optimization problem is equivalent to finding (near) maximal configurations of a Boltzmann machine. The parallelism of a Boltzmann machine can be used to speed up the computations and, thus Boltzmann machines can be viewed as a massively parallel approach to simulated annealing [Aarts & Korst, 1989a; 1991].

**General strategy**

An instance of a combinatorial optimization problem is represented by a 3-tuples \((\mathcal{S}, \mathcal{S}', f)\), where \(\mathcal{S}\) denotes the finite set of solutions, \(\mathcal{S}' \subseteq \mathcal{S}\) the set of feasible solutions, satisfying the constraints in the problem, and \(f : \mathcal{S} \to \mathbb{R}\) the cost function that assigns a real number to each solution; see also Section 2. Now, the problem is to find a feasible solution for which the cost function is optimal.

As already mentioned in Section 2.1, solutions can be represented as finite sets of discrete variables \(\mathcal{X} = \{x_1, \ldots, x_n\}\). So, \(\mathcal{S} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_n\), where \(\mathcal{X}_i, i = 1, \ldots, n\), is a finite set denoting the domain of the variable \(x_i\).

To use a Boltzmann machine in combinatorial optimization, a bijective function \(m : \mathcal{R} \to \mathcal{S}\) is defined which maps the set of configurations \(\mathcal{R}\) onto the set of solutions \(\mathcal{S}\), by applying the following general strategy.

- Formulate the optimization problem as a 0-1 programming problem, i.e., formulate the problem such that \(\mathcal{X}_i = \{0, 1\}\), for \(i = 1, \ldots, n\).
- Define a Boltzmann machine \(\mathcal{B} = (U, C)\), such that the state of each unit \(u_i \in U = \{u_1, \ldots, u_n\}\) determines the value of variable \(x_i\). If unit \(u_i\) is 'on' then \(x_i = 1\); if \(u_i\) is
'off' then \( x_i = 0 \). Thus, \( x_i = k(u_i) \). Clearly, this defines a bijective function \( m \) which maps the set of configurations \( \mathcal{R} \) onto the set of solutions \( \mathcal{S} \).

- Define the set of connections \( \mathcal{C} \) and the corresponding strengths such that the consensus function \( C \) is feasible and order-preserving.

The following definitions explain what we mean by a feasible and order-preserving consensus function.

**Definition 4.17 (feasibility)** Let \( B \) be a Boltzmann machine that implements a combinatorial optimization problem \( (\mathcal{S}, \mathcal{S}', f) \) and let \( m \) be a bijective mapping from \( \mathcal{R} \) onto \( \mathcal{S} \), then the consensus function \( C \) of the Boltzmann machine is called feasible if all local maxima of the consensus function \( C \) correspond to feasible solutions of the optimization problem, i.e.,

\[
m(\hat{\mathcal{R}}) \subseteq \mathcal{S}',
\]

where \( \hat{\mathcal{R}} \) denotes the set of locally maximal configurations and \( m(\hat{\mathcal{R}}) = \{ m(k) \mid k \in \hat{\mathcal{R}} \} \).

**Definition 4.18 (order-preservingness)** Let \( B \) be a Boltzmann machine that implements a minimization problem \( (\mathcal{S}, \mathcal{S}', f) \) and let \( m \) be a bijective mapping from \( \mathcal{R} \) onto \( \mathcal{S} \), then the consensus function \( C \) of the Boltzmann machine is called order-preserving if for all \( k, l \in \mathcal{R} \), with \( m(k), m(l) \in S' \), we have

\[
f(m(k)) < f(m(l)) \Rightarrow C_k > C_l,
\]

and equivalently for maximization.

Feasibility of the consensus function implies that a feasible solution is found, since a Boltzmann machine always converges to a configuration \( k \in \hat{\mathcal{R}} \). Order-preserving means that the order with respect to cost of the solutions in the minimization problem and the order with respect to consensus of the configurations in the corresponding Boltzmann machine are opposit. So, if the consensus function is feasible and order-preserving, then the consensus is maximal for configurations corresponding to an optimal (feasible) solution. Furthermore, under the same conditions, near-optimal local maxima of the consensus function correspond to near-optimal feasible solutions. This property is quite important, since it facilitates the use of Boltzmann machines for approximation purposes.

To illustrate the use of Boltzmann machines in combinatorial optimization we discuss two different NP-hard problems, i.e., INDEPENDENT SET and UNIFORM GRAPH PARTITIONING. INDEPENDENT SET is chosen for its simplicity with respect to the mapping onto Boltzmann machines. UNIFORM GRAPH PARTITIONING is chosen to illustrate some of the technical steps involved in more difficult mappings.

**Definition 4.19 (INDEPENDENT SET)** Given a graph \( G = (V, E) \), find an independent set of maximal size, i.e., a subset \( V' \subseteq V \), such that for all \( v_i, v_j \in V' \) the edge \( \{v_i, v_j\} \) is not in \( E \), and such that \( |V'| \) is maximal.

In a 0-1 programming formulation, INDEPENDENT SET can be written as follows. Let the 0-1 variables \( x_i \) be defined by

\[
x_i = \begin{cases} 
1 & \text{if } v_i \in V' \\
0 & \text{if } v_i \notin V', 
\end{cases}
\]
then INDEPENDENT SET can be written as follows.

\[
\begin{align*}
\text{maximize} & \quad f(X) = \sum_{i=1}^{n} x_i, \\
\text{subject to} & \quad x_i x_j e_{ij} = 0, \quad i, j = 1, \ldots, n,
\end{align*}
\]  \hspace{1cm} (63)

where \( n = |V| \) and \( e_{ij} \) is defined as

\[
e_{ij} = \begin{cases} 
1 & \text{if } \{v_i, v_j\} \in E \\
0 & \text{if } \{v_i, v_j\} \notin E.
\end{cases}
\]

The problem can now be mapped onto a Boltzmann machine \( B = (U, C) \) as follows. For each variable \( x_i \) a unit \( u_i \in U \) is defined. To guarantee that no adjacent vertices \( i \) and \( j \) are in \( V' \), inhibitory connections are introduced which connect the corresponding units \( u_i \) and \( u_j \). Furthermore, bias connections are used to give each unit a tendency to be 'on'. Hence, the set of connections \( C \) is chosen as the union of the following two disjoint sets, where \( u_i, u_j \in U \), \( 1 \leq i \leq n \).

- **Bias connections.**  \( C_b = \{ \{u_i, u_i\} \mid v_i \in V \} \).

- **Inhibitory connections.**  \( C_h = \{ \{u_i, u_j\} \mid \{v_i, v_j\} \in E \} \).

The following theorem yields values for the connection strengths for which the corresponding consensus function is feasible and order-preserving [Aarts & Korst, 1989a].

**Theorem 4.7** Let \( \beta \) be a positive number and let

\[
\forall \{u_i, u_i\} \in C_b : \quad s_{(u_i, u_i)} = \beta, \quad \text{and} \quad \\
\forall \{u_i, u_j\} \in C_h : \quad s_{(u_i, u_j)} < -\beta,
\]

then the consensus function \( C \) for INDEPENDENT SET is feasible and order-preserving.

To prove the theorem the set of configurations is partitioned into two subsets: \( R = R_A \cup R_B \) with \( R_A \cap R_B = \emptyset \), where \( R_A \) and \( R_B \) denote the sets of configurations corresponding to feasible and infeasible solutions, respectively. Now, feasibility follows directly since it is easy to show that

\[
\forall k \in R_B \quad \exists u_i \in U : \quad \Delta C_k(u_i) > 0.
\]

The order-preserving requirement follows directly from the fact that for a feasible solution \( k \in R_A \), the consensus function can be written as

\[
C_k = \sum_{\{u_i, u_i\} \in C_b} s_{(u_i, u_i)} k^2(u_i) \\
= \sum_{u_i \in U} \beta k(u_i) \\
\sum_{i \in V'} \beta x_i.
\]

61
which differs only with a fixed multiple $\beta$ from (63). Clearly, the consensus function and the cost function are identical if $\beta$ is chosen equal to 1.

As an example, we discuss the estimation of the value of the constant $\alpha_c$ of (61) for INDEPENDENT SET.

Let $V^*$ denote the maximum independent set. One then can verify that $C^* = |V^*|$, $C^+ = |V^*| - 1$ and $S^* = S_0 = |V|$. Excluding disconnected graphs, which trivially implies that $2|V^*| \leq |V|$, (61) yields the following critical value.

$$\frac{2}{|V| + 2 - 2|V^*|} \leq \alpha_c \leq \frac{2|V^*|}{|V|}.$$  \hspace{1cm} (64)

The bounds in (64) coincide if and only if $|V^*| = 1$ or $2|V^*| = |V|$. To give an indication of the value of $\alpha_c$, consider random graphs with a given number of vertices and edges chosen with a probability equal to $d/|V| - 1$, implying that the expected degree of each vertex equals $d$. For this class of graphs the average size of an independent set equals $|V|/d$, whereas maximum independent sets are on the average three times as large as the average independent sets [Bollobás, 1985]. Hence $1/d \leq \alpha_c \leq 6/d$.

Consider UNIFORM GRAPH PARTITIONING; see Definition 2.7. To map this problem onto a Boltzmann machine, we first reformulate it as a quadratic assignment problem as follows. The assignment of vertices in $V$ to the subsets $V_1$ and $V_2$ is modelled by a bijective function $f : V \rightarrow \{1, \ldots, 2n\}$. By definition, a vertex $v_i$ is assigned to subset $V_1$ if and only if $f(v_i) \leq n$. We can thus give the following 0-1 formulation. Let 0-1 variable $x_{ik}, 1 \leq i, k \leq 2n$, be defined by

$$x_{ik} = \begin{cases} 
1 & \text{if } f(v_i) = k \\
0 & \text{if } f(v_i) \neq k.
\end{cases}$$

Furthermore, let $w_{ij}$ be the weight of the edge between vertices $v_i$ and $v_j$. By definition, $w_{ij} = 0$ if there is no edge between $v_i$ and $v_j$. We can now formulate the problem as follows.

$$\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{2n} \sum_{k=1}^{2n} \sum_{j=1}^{n+1} w_{ij} x_{ik} x_{jl}, \\
\text{subject to} & \quad \sum_{k=1}^{2n} x_{ik} = 1, \quad \text{for } k = 1, \ldots, 2n, \text{ and} \\
& \quad \sum_{i=1}^{2n} x_{ik} = 1, \quad \text{for } i = 1, \ldots, 2n.
\end{align*}$$  \hspace{1cm} (65)

The problem can now be mapped onto a Boltzmann machine $B = (\mathcal{U}, \mathcal{C})$ as follows. For each 0-1 variable $x_{ik}$ we introduce a unit $u_{ik} \in \mathcal{U}$ that is 'on' if $x_{ik} = 1$. The set of connections $\mathcal{C}$ is taken as the union of the following disjoint sets, where $u_{ik}, u_{jl} \in \mathcal{U}, 1 \leq i, j, k, l \leq 2n$.

- **Bias connections.** $C_b = \{\{u_{ik}, u_{ik}\}\}$.
- **Inhibitory connections.** $C_h = \{\{u_{ik}, u_{jl}\} | (i = j) \text{ or } (k = l)\}$.
- Edge connections. \[ C_e = \{ \{u_{ik}, u_{jl}\} \mid (v_i, v_j) \in E \text{ and } c(k, l) = 1\} \], with
\[
c(k, l) = \begin{cases} 1 & \text{if } (k \leq n < l) \text{ or } (l \leq n < k) \\ 0 & \text{otherwise}. \end{cases}
\]

We can now state the following theorem.

**Theorem 4.8** Let \( \varepsilon \) be a small positive number and let
\[
\begin{align*}
s_{u_{ik}, u_{ik}} &= \sum_{j=1}^{2n} w_{ij} + \varepsilon, & \text{for all } \{u_{ik}, u_{ik}\} \in C_b, \\
s_{u_{ik}, u_{jl}} &= -\max(s_{u_{ik}, u_{ik}}, s_{u_{jl}, u_{jl}}) - \varepsilon, & \text{for all } \{u_{ik}, u_{jl}\} \in C_h, \text{ and} \\
s_{u_{ik}, u_{jl}} &= -w_{ij}, & \text{for all } \{u_{ik}, u_{jl}\} \in C_c,
\end{align*}
\]
then the consensus function \( C \) for UNIFORM GRAPH PARTITIONING is feasible and order-preserving.

Given that \(|V| = 2n\) and \(|E| = m\), we observe that the above mapping requires \(4n^2\) units and \(2n^2(2+8n+m)\) connections. We could next try to reduce the number of units and connections.

It is possible to reduce this number of units and connections by relaxing the requirement that \(V_1\) and \(V_2\) have exactly the same number of vertices. This can be realized by simply dropping this constraint and adding an extra term to the cost function of (65) that penalizes the imbalance in the cardinalities of \(V_1\) and \(V_2\). We next show that in that case we only have to associate one unit with each vertex \(v_i \in V\), instead of the \(4n^2\) units in the case of a strictly uniform partition.

We associate with each vertex \(v_i \in V\) the 0-1 variables \(x_i\) that is defined as follows.

\[
x_i = \begin{cases} 1 & \text{if } v_i \in V_1 \\ 0 & \text{if } v_i \in V_2. \end{cases}
\]

The cost function of (65) can then be replaced by the following expression.
\[
\sum_{i=1}^{2n} \sum_{j=1}^{2n} w_{ij}x_i(1 - x_j) + \lambda \left(|V_1| - |V_2| \right)^2,
\]
where
\[
(|V_1| - |V_2|)^2 = \left( \sum_{i=1}^{2n} x_i - \sum_{j=1}^{2n} (1 - x_j) \right)^2,
\]
and \(\lambda \in \mathbb{R}^+\) is a weight factor that determines the amount by which the imbalance in the cardinalities of \(V_1\) and \(V_2\) is penalized.

The cost function of (66) can be rewritten as
\[
\sum_{i=1}^{2n} \sum_{j=1}^{2n} (w_{ij} - 4\lambda)x_i(1 - x_j) + 4\lambda n^2.
\]
Discarding the constant $4\lambda n^2$ and using the fact that $w_{ii} = 0$, we obtain the following cost function, which is to be minimized without further constraints.

$$-\sum_{i=1}^{2n} \sum_{j=1, j \neq i}^{2n} (w_{ij} - 4\lambda)x_ix_j + \sum_{i=1}^{2n} \sum_{j=1, j \neq i}^{2n} (w_{ij} - 4\lambda)x_i.$$  

This problem can be straightforwardly mapped onto a Boltzmann machine $B = (U, C)$ as follows. For each $x_i$ we introduce a unit $u_i \in U$, where the unit is defined to be 'on' iff the corresponding variable is 1. The set of connections $C$ is taken as the union of the following two disjoint sets, where $u_i, u_j \in U$ and $1 \leq i, j \leq 2n$.

- **Bias connections.** $C_b = \{\{u_i, u_i\}\}$.
- **Edge connections.** $C_e = \{\{u_i, u_j\} \mid i \neq j\}$.

We can now prove the following theorem.

**Theorem 4.9** Let $\lambda \in \mathbb{R}^+$ be a weighting factor, and let

$$s_{\{u_i, u_i\}} = -\sum_{j=1, j \neq i}^{2n} (w_{ij} - 4\lambda), \quad \text{for all } \{u_i, u_i\} \in C_b, \text{ and}$$

$$s_{\{u_i, u_j\}} = w_{ij} - 4\lambda, \quad \text{for all } \{u_i, u_j\} \in C_e.$$  

Then the consensus function $C$ for the relaxed version of UNIFORM GRAPH PARTITIONING is feasible and order-preserving.

Evidently, we now have a mapping that only requires $n$ units and $n^2$ connections.

**Performance**

The feasibility of the model of Boltzmann machines for solving (approximately) combinatorial optimization problems has been investigated for a number of problems including MAXCUT, INDEPENDENT SET, and GRAPH COLOURING [Korst & Aarts, 1989], MATCHING and GRAPH PARTITIONING [Aarts & Korst, 1991] and TRAVELLING SALESMAN [Aarts & Korst, 1989b]. For a number of these problems the investigations are based on computer simulations comparing the performance with simulated annealing. Roughly speaking the following conclusions are obtained.

For MAXCUT, INDEPENDENT SET, and GRAPH COLOURING, the Boltzmann machine performs very well, i.e. it converges quickly to near-optimal results. The Boltzmann machine can find final solution comparable, in quality, to the solutions obtained by sequential simulated annealing. Running times are low-order polynomials in the sizes of the instances, typically between $|V|^{1.1}$ and $|V|^{2.0}$, where $|V|$ denotes the number of vertices in the instance graphs. For all three problems the Boltzmann machines obtain a linear speed-up compared to the sequential simulated annealing implementations.

For TRAVELLING SALESMAN results are poor so far. It is concluded that it is hard to obtain near-optimal results, both when implemented as a quadratic or as a linear assignment problem. The convergence of the Boltzmann machine is very slow and the quality of the final results is poor when compared to the solutions obtained by sequential simulated annealing. This rather
disappointing performance is predominantly due to the 0-1 programming formulations that were used. Wilson & Pawley [1988] have studied the performance of Hopfield networks for TRAVELLING SALESMAN using the quadratic assignment formulation. They also concluded that the lack of performance they observed in their simulations is partly due to the use of this formulation. It is, however, no simple matter to devise other formulations that would do better.

As already mentioned, the importance of Boltzmann machines for combinatorial optimization lies in its significance as a massively parallel approach to simulated annealing. Simulations support this significance since a substantial speed up can be achieved by implementations on multiprocessor systems. The significance is further increased when Boltzmann machines are implemented on special-purpose hardware or general neurocomputers. In this way the annealing process can be performed extremely fast by using analog devices for instance, which add up the incoming charge and make the stochastic decisions by means of noise. The design of hardwired networks has been the subject of study for some time. Alspector & Allen [1987] have presented a design of a VLSI chip with $5 \cdot 10^5$ gates, implementing a Boltzmann machine consisting of approximately 2000 units. They estimate that their chip will run about a million times faster than simulations on a VAX. Optical implementations of the Boltzmann machine as the one proposed by Ticknor & Barrett [1987] or Lalanne, Rodier, Richard, Chavel, Belhaire, Madani & Garda [1990] are claimed to increase this factor even further.

4.6 Related topics

As for local search and simulated annealing, the literature presents a number of developments that are related in some sense to the Boltzmann machine. Below we give a few examples.

Smolensky [1986] proposes a harmonium network model to handle problem solving issues in parallel cognitive models. The model is developed independently from the Boltzmann machines but shows some interesting correspondences, since it is also inspired by the use of statistical physics in computational networks. It uses a representational network with a node set consisting of knowledge atoms and feature nodes. The knowledge atoms have activation values and the feature nodes have feature values. Furthermore, there is a harmony function that gives a measure of the consistency between a set of activated rule nodes and a set of activated circuit nodes. The objective is to maximize the harmony function. Again a transition mechanism is used that accepts node transitions with a probability that is given by an exponential function of the difference in harmony between the two states and a control parameter. The relation with Boltzmann machines is obvious: the semantic network is analogous to structural network, the harmony function to the consensus function, and in both cases we have a randomized state transition mechanism controlled by a dummy parameter.

Another interesting relation is based on the analogy between Boltzmann machines and spin glass models in thermal physics [Morgenstern, 1987; Toulouse, Dehaene & Changeux, 1986]. In such a spin glass model one considers an ensemble of spin particles, having spin 'up' or 'down', i.e. $S_i = \pm 1$. The corresponding Hamiltonian is given by

$$H = -\frac{1}{2} \sum_{i<j} J_{ij} S_i S_j + h_0 \sum_i S_i,$$
where the $J_{ij}$ denote the coupling strengths of the short-range spin-spin interaction between two particles $i$ and $j$, and $h_0$ denotes the strength of an external magnetic field.

The analogy with the Boltzmann machine is again evident. The spins play the role of the states of the individual units and the Hamiltonian is the equivalent of the consensus function. The spin glass model described above is characteristic for systems studied in statistical physics. Hence, if the analogy with neural networks can be pursued far enough, it permits the introduction into the theory of neural networks of such concepts as coupled differential equations, phase transitions, correlation functions, mean field theories, etc., which provides a powerful tool for studying the continuous-time behaviour of neural networks; see for instance Peretto [1984].

As an interesting example we briefly discuss the *mean field annealing* approach to Boltzmann machines [Van den Bout & Miller III, 1990; Peterson & Söderberg, 1989]. Given a Boltzmann machine with a stationary distribution given by (47). The probability of a unit $u \in \mathcal{U}$ to be in one of the two states "0" or "1" is then given by

$$
\mathbb{P}_c\{k(u) = 0\} = \frac{\sum_{k \in k(u) = 0} \exp(C_k/c)}{\sum_{i \in \mathcal{R}} \exp(C_i/c)}, \quad \text{and} \quad (67)
$$

$$
\mathbb{P}_c\{k(u) = 1\} = \frac{\sum_{k \in k(u) = 1} \exp(C_k/c)}{\sum_{i \in \mathcal{R}} \exp(C_i/c)}. \quad (68)
$$

This equilibrium state can be modelled from a different point of view using concepts from mathematical statistics. To this end we define

$$
\mathbb{E}_c(C) = \sum_{\{v,w\} \in \mathcal{C}} s_{\{v,w\}} \mathbb{E}_c(k(v)k(w)) \quad (69)
$$

as the expected value of the consensus for a given value of the control parameter $c$. Furthermore, define $C_0(u,c) = \mathbb{E}_c(C|k(u) = 0)$ and $C_1(u,c) = \mathbb{E}_c(C|k(u) = 1)$ as the expected values of the consensus experienced by unit $u$ in equilibrium at $c$ in states "0" and "1", respectively. In other words, from the point of view of unit $u$, the Boltzmann machine can be in two equilibrium states, one with expected consensus $C_0(u,c)$ and the other with expected consensus $C_1(u,c)$. Thus the probabilities of (67) and (68) can be approximated by

$$
\mathbb{P}_c\{k(u) = 0\} = \frac{\exp(C_0(u,c)/c)}{\exp(C_0(u,c)/c) + \exp(C_1(u,c)/c)}, \quad \text{and}
$$

$$
\mathbb{P}_c\{k(u) = 1\} = \frac{\exp(C_1(u,c)/c)}{\exp(C_0(u,c)/c) + \exp(C_1(u,c)/c)}.
$$

The idea now is to model equilibrium from the point of view of the expected consensus or *mean field* experienced by a unit. The next step is to approximate (69) by

$$
\mathbb{E}_c(C) = \sum_{\{u,w\} \in \mathcal{C}} s_{\{u,w\}} \mathbb{E}_c(k(v)) \mathbb{E}_c(k(w)).
$$
which is known as the mean field approximation. The approximation is based on the assumption
that, despite the fact that the states of two units $v$ and $w$ are dependent, the correlated con­
tribution to the expected consensus is approximately zero. From this approximation, we obtain
the following set of equations.

$$
\frac{\partial \mathbb{E}_c(C)}{\partial \mathbb{E}_c(k(u))} = \sum_{\{u,w\} \in C} a_{\{u,w\}} \mathbb{E}_c(k(w)) = C_1(u, c) - C_0(u, c) = \Phi(u, c), \quad \text{and} \quad (70)
$$

$$
\mathbb{E}_c(k(u)) = \frac{1}{1 + \exp(-\Phi(u, c)/c)}. \quad (71)
$$

So the dynamics of the Boltzmann machine is completely determined by (70) and (71).

For practical purposes, the expectations $\mathbb{E}(k(u))$ are approximated by the average values
$\overline{k}(u)$. Consequently, a stable configuration with maximal consensus can be reached by starting at a
large value of $c$, where $\mathbb{E}_c(k(u)) = \frac{1}{2}$ for all $u \in \mathcal{U}$, and lowering $c$ slowly under the condition
that (70) and (71) hold. This leads to the following scheme.

**Step 1.** Initialize the states: $\overline{k}(u) = \frac{1}{2} + \delta$, for all $u \in \mathcal{U}$ and small value of $\delta$, and choose an
initial value of the control parameter $c$. The $\delta$ is added to break the symmetry.

**Step 2.** Select a unit $u \in \mathcal{U}$ and calculate

$$
\Psi = \sum_{\{u,w\} \in C} a_{\{u,w\}} \overline{k}(w), \quad \text{and}
$$

$$
\overline{k}(u) = \frac{1}{\exp(-\Psi/c)}.
$$

**Step 3.** Repeat Step 2 a number of times until a fixed point is reached.

**Step 4.** Lower $c$ and repeat Steps 2 and 3.

This algorithm requires a similar set of parameters as in a Boltzmann machine or simulated
annealing cooling schedule: initial and final values of $c$, a decrement function, and a number of
trials at each value of $c$ to reach the fix point. For more details we refer to Van den Bout &
Miller III [1990] and Peterson & Söderberg [1989].

The algorithm outlined above, and parallel versions of it, have been applied to GRAPH PARTI­
TIONING and TRAVELLING SALESMAN. [Van den Bout & Miller III, 1990; Peterson & Söderberg,
1989]. The authors report that substantial speed up of convergence can be achieved compared
to the classical approach.
5 Conclusion

In this chapter we have surveyed work on deterministic and randomized local search. We have been concentrating on theoretical aspects of local search with simulated annealing and Boltzmann machines as the central issues. As we have seen in the sections on related topics, there is much more to the subject than we were able to cover in this chapter. In this concluding section we elaborate on some of these items from a more conceptual point of view, trying to formulate some points of interest.

As was already mentioned in Section 2.5, there exists a large variety of search algorithms based on the exploration of neighbourhoods. The best-known examples are simulated annealing, tabu search and genetic algorithms. Each of these algorithms has become well-established with its own dedicated group of practitioners. More recently people have been concentrating on the design of all kinds of hybrid algorithms consisting of extensions and combinations of the above mentioned algorithms; see for instance the proceedings of the first conference on Parallel Problem Solving by Nature [Schwefel & Manner, 1990]. This has led to a cluttered collection of algorithms, which calls for a structured approach that generalizes the common features into a generic framework. As already mentioned, a first approach to such a framework is given by Vaessens, Aarts & Lenstra [1992], who presented a template that captures most of the common features in a generic local search approach. However, what still needs to be done is to study convergence properties of such a generic approach and to provide measures that enable a comparative performance evaluation of the various algorithms.

The broad interest in local search evidently stems from its significant practical success. Here, we see two interesting new developments.

First, there is the use of local search to find good solutions for extremely large instances of practical optimization problems. For a number of problems, local search seems to be the only approach that can handle large instances both efficiently and effectively. For instance, there are industrially relevant problem instances of TRAVELING SALESMAN with up to 100,000 cities, that can only be successfully handled with local search algorithms; see for instance [Bentley, Johnson, McGeoch & Rothberg, 1990; Johnson, 1990b; Bland & Shallcross, 1989; Reinelt, 1992]. An other example is the placement of several thousands of so-called standard cells in chip layouts, which so far can only be effectively done by local search algorithms; see Shahookar & Mazumder [1991]. Such a development calls for the search of sophisticated data structures and dynamic techniques that allow the exploration of very large neighbourhoods.

Secondly, the success on standard problems such as TRAVELING SALESMAN, GRAPH PARTITIONING, and JOB SHOP SCHEDULING raises the question whether local search can be successfully applied to generalized models such as vehicle routing, resource constrained project scheduling and all sorts of clustering problems. The complexity of these models so far has hindered the design of effective algorithms. However, it should be possible to think of good neighbourhoods for the problems in these fields. Examples have been reported for vehicle routing with time windows by Savelsbergh [1990], for a generalized job shop scheduling problem by Widmer [1991], and for a practical clustering problem by Brown & Huntley [1992].
Since its introduction in 1983, simulated annealing has been extensively studied from a theoretical point of view and it has been successfully applied to problems in many different areas. We already mentioned several overviews; see Section 3.5. Concentrating on the theoretical results, we conclude that the asymptotic behaviour of the algorithm is well understood and that there is little need for further investigations on this point. What is still open is the finite-time behaviour of the algorithm. It would be of great interest to have more insight in the outcome of simulated annealing if the algorithm is allowed only a finite running time. So far most results on this issue have been obtained with empirical studies. There are however also some interesting theoretical approaches. We mentioned the following ones.

Strenski & Kirkpatrick [1991] have studied optimal finite-length annealing schedules. By using the master equation, they computed the exact distribution of outcomes of the annealing algorithm for extremely small problem instances. In this way the performance of annealing schedules can be theoretically analysed. For a highly symmetric instance of GRAPH PARTITIONING, Strenski & Kirkpatrick [1991] observed that optimal annealing schedules need not be monotonically decreasing in the control parameter. More specifically, for their example they showed that optimal annealing schedules exist consisting of a sequence of control parameter values with a number of zero's at the beginning.

Unfortunately, the analysis of Strenski & Kirkpatrick [1991] is restricted to extremely small problem instances. Performance analyses of simulated annealing for larger problem instances have been based on explorations of neighbourhood graphs. For instance, Kirkpatrick & Toils have defined hierarchical structures in the set of solutions, which they call ultrametric structures. The presence or absence of these structures in instances of combinatorial optimization problems has been associated with the performance of simulated annealing. They demonstrated ultrametric structures in some instances of graph partitioning and TRAVELING SALESMAN. Furthermore, they conjectured that simulated annealing would perform better on problem instances with ultrametric structures than on those without. A similar and more rigorous approach was presented by Sorkin [1991], who analysed the finite-time behaviour of simulated annealing on problem instances with fractal neighbourhood structures. More precisely he showed that certain fractal properties of the neighbourhood structure and some weak scaling properties of the cost function are sufficient conditions for simulated annealing to find solutions with an expected cost deviating no more than $\epsilon$, $\epsilon \in \mathbb{R}^+$, from the optimal value, within a running time that can be bounded by a polynomial in $\epsilon^{-1}$ and order given by parameters of the fractal. An interesting corollary of this work is that for a number of placement problems in VLSI design it is possible to give neighbourhood structures that are fractal. This would explain the success of simulated annealing in this problem area.

A different approach to the finite-time performance of annealing is given by Van Laarhoven, Boender, Aarts & Rinnooy Kan [1989]. These authors have used a Bayesian approach to analyse the distribution of outcomes of a simulated annealing algorithm. It was shown that it is possible to accurately predict the distribution of outcomes of a subsequent Markov chain in the simulated annealing algorithm. It was however not possible to accurately predict the deviation of the final solution from the optimum value.

In this chapter we have discussed the role of neural networks in local search from the point of
view of speeding up the calculations by using the inherent parallelism of these networks. As we have seen in Section 4.5, the use of neural networks in combinatorial optimization imposes conditions on the problem representations that are used. As we already pointed out, for some problems these conditions are not restrictive, e.g. problems on graphs, whereas for others they are quite restrictive as for instance in TRAVELING SALESMAN. So, in addition to the advantage of a considerable gain in computation time, neural networks may also suffer from a lack of effectiveness as a result of representation limitations.

So far the practical success of neural networks in combinatorial optimization is only minor [Looi, 1992]. Besides the graph theoretical problems mentioned above, scheduling is one of the few practical problem areas for which satisfactory results have been reported [Foo & Takefuji, 1988; Van Hulle, 1991; Zhou, Cherkassky, Baldwin & Olson, 1991]. In fact, parallel search is the only capability of neural networks that is used in these applications. Other capabilities, such as association and learning, which play a dominant role in the successes of neural networks in other application areas such as pattern recognition, have been largely discarded so far in combinatorial optimization. These capabilities could be successfully used in those combinatorial optimization problems, where part of the input data is not known or in which assumptions must be made on certain quantities. An example of such a problem area is production planning. For instance, Zwietering, Van Kraaij, Aarts & Wessels [1991] have reported on the application of neural networks to the rolling horizon version of the dynamic lot sizing problem. They used the associative and learning capabilities in addition to parallel search to deal with the changes in the demand over the periods of the rolling horizon. This approach is quite simple and turns out to be superior to the existing methods. It has opened new perspectives on the practical use of neural networks in this specific area of combinatorial optimization.
Bibliography


Chvatal, V. [1983], Linear Programming, Freeman, San Francisco.


Cook, S. A. [1972], An overview of computational complexity, Communications of the ACM 26, 400-408.


Dantzig, G. B. [1963], Linear Programming and Extensions, Princeton University Press, Princeton.

Dueck, G. [1989], New optimization heuristics: the great deluge algorithm and the record-to-travel, manuscript, IBM Germany, Heidelberg Scientific Centre, Heidelberg.


Fiduccia, C. M., and R. M. Mattheyses [1982], A linear-time heuristic for improving network par-


French, S. [1982], Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop, Horwood, Chichester.


Glover, F. [1991], Simple tabu thresholding in optimization, manuscript, Graduate School of Business, University of Colorado, Boulder.


Greening, D.R. [1990], Parallel simulated annealing techniques, Physica D 42, 293-306.


Harary, F. [1972], Graph Theory, Addison-Wesley, Reading.


Held, M., and R.M. Karp [1971], The traveling salesman problem and minimum spanning trees: Part II, Mathematical Programming 1, 6-25.

Hertz, A., and D. De Werra [1991], The tabu search metaheuristic: how we used it, Annals of Mathematics and Artificial Intelligence 1, 111-121.


Holley, R., and D. Stroock [1988], Simulated annealing via Sobolev inequalities, Communications in Mathematical Physics 115, 553-569.


Huang, M.D., F. Romeo, and A. Sangiovanni-Vincentelli [1986], An efficient general cooling schedule for simulated annealing, Proceedings of the IEEE International Conference on Comput-


Local optimization and the traveling salesman problem, *Springer Lecture Notes in Computer Science* 447, 446-461.


How easy is local search?, *Journal of Computer and System Science* 37, 79-100.


Optimal acceptance probability for simulated annealing, *Stochastics and Stochastics Reports* 29, 221-226.


Combinatorial optimization on a Boltzmann machine, *Journal of Parallel and Distributed Computing* 6, 331-357.


Theoretical and Computational Aspects of Simulated Annealing, PhD. thesis, Erasmus University Rotterdam, Rotterdam.


LUEKER, G. [1975], Two NP-complete problems in nonnegative integer programming, manuscript, TR 178, Princeton University, Princeton.


O’REIGERTAIGH, M., J.K. LENSTRA, A.H.G. RINNOOY KAN (EDS.) [1985], Combinatorial


PARBERRY, I., AND G. SCHNITGER [1989], Relating Boltzmann machines to conventional models of computation, Neural Networks 2, 59-67.


ROY, B., AND B. SUSSMANN [1964], Les problèmes d’ordonnancement avec constraints disjonctives, Note DS No. 9 bis, SEMA, Paris, France.


SCHUR, P. C. [1989], Classification of acceptance criteria for the simulated annealing algorithm, Memorandum COSOR 89-29, Eindhoven University of Technology, Eindhoven.


SHAHOOKAR, K., AND P. MAZUMDER [1991], VLSI cell placement techniques, Computing Surveys 23, 143-220.

SORKIN, G.B. [1991], Efficient simulated annealing on fractal energy landscapes, Algorithmica 6, 367-418.


YANNAKAKIS, M. [1990], The analysis of local search problems and their heuristics, Lecture Notes in Computer Science 415, 298-310.


In this series appeared:

91/01 D. Alstein

91/02 R.P. Nederpelt
Implication. A survey of the different logical analyses "if...then...", p. 26.
H.C.M. de Swart

91/03 J.P. Katoen
Parallel Programs for the Recognition of $P$-invariant Segments, p. 16.
L.A.M. Schoenmakers

91/04 E. v.d. Sluis
Performance Analysis of VLSI Programs, p. 31.
A.F. v.d. Stappen

91/05 D. de Reus
An Implementation Model for GOOD, p. 18.

91/06 K.M. van Hee
SPECIFICATIEMETHODEN, een overzicht, p. 20.

91/07 E. Poll
CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.

91/08 H. Schepers
Terminology and Paradigms for Fault Tolerance, p. 25.

91/09 W.M.P.v.d.Aalst
Interval Timed Petri Nets and their analysis, p. 53.

91/10 R.C. Backhouse
POLYNOMIAL RELATORS, p. 52.
P.J. de Bruin

91/11 R.C. Backhouse
Relational Catamorphism, p. 31.
P.J. de Bruin
G. Malcolm
E. Voermans
J. v.d. Woude

91/12 E. van der Sluis

91/13 F. Rietman
A note on Extensionality, p. 21.

91/14 P. Lemmens
The PDB Hypermedia Package. Why and how it was built, p. 63.

91/15 A.T.M. Aerts
K.M. van Hee

91/16 A.J.J.M. Marcelis
An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.

91/17 A.T.M. Aerts
Transforming Functional Database Schemes to Relational Representations, p. 21.
P.M.E. de Bra
K.M. van Hee
91/18 Rik van Geldrop
Transformational Query Solving, p. 35.

91/19 Erik Poll
Some categorical properties for a model for second order
lambda calculus with subtyping, p. 21.

91/20 A.E. Eiben
R.V. Schuwer

91/21 J. Coenen
Assertional Data Reification Proofs: Survey and
W.-P. de Roever
Perspective, p. 18.
J.Zwiers

91/22 G. Wolf
Schedule Management: an Object Oriented Approach, p.

91/23 K.M. van Hee
Z and high level Petri nets, p. 16.
L.J. Somers
M. Voorhoeve

91/24 A.T.M. Aerts
Formal semantics for BRM with examples, p. 25.
D. de Reus

91/25 P. Zhou
A compositional proof system for real-time systems based
J. Hooman
on explicit clock temporal logic: soundness and complete
R. Kuiper
ness, p. 52.

91/26 P. de Bra
The GOOD based hypertext reference model, p. 12.
G.J. Houben
J. Paredaens

91/27 F. de Boer
Embedding as a tool for language comparison: On the
C. Palamidessi
CSP hierarchy, p. 17.

91/28 F. de Boer
A compositional proof system for dynamic process
creation, p. 24.

91/29 H. Ten Eikelder
Correctness of Acceptor Schemes for Regular Languages,
R. van Geldrop
p. 31.

91/30 J.C.M. Baeten
An Algebra for Process Creation, p. 29.
F.W. Vaandrager

91/31 H. ten Eikelder
Some algorithms to decide the equivalence of recursive

91/32 P. Struik
Techniques for designing efficient parallel programs, p.

91/33 W. v.d. Aalst
The modelling and analysis of queueing systems with
QNM-ExSpect, p. 23.

91/34 J. Coenen
Specifying fault tolerant programs in deontic logic,

91/35 F.S. de Boer
Asynchronous communication in process algebra, p. 20.
J.W. Klop
C. Palamidessi
A note on compositional refinement, p. 27.

A compositional semantics for fault tolerant real-time systems, p. 18.

Real space process algebra, p. 42.

Program derivation in acyclic graphs and related problems, p. 90.

Conservative fixpoint functions on a graph, p. 25.

Discrete time process algebra, p. 45.

The fine-structure of lambda calculus, p. 110.

On stepwise explicit substitution, p. 30.


Composition and decomposition in a CPN model, p. 55.

Demonic operators and monotype factors, p. 29.


Set theory and nominalisation, Part II, p. 22.

The total order assumption, p. 10.

A system at the cross-roads of functional and logic programming, p. 36.

Integrity checking in deductive databases; an exposition, p. 32.

Interval timed coloured Petri nets and their analysis, p. 20.

A unified approach to Type Theory through a refined lambda-calculus, p. 30.

Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, p. 36.

Are Types for Natural Language? P. 32.

Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.
<table>
<thead>
<tr>
<th>Date</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>92/22</td>
<td>R. Nederpelt, F. Kamareddine</td>
<td>A useful lambda notation, p. 17.</td>
</tr>
<tr>
<td>92/23</td>
<td>F. Kamareddine, E. Klein</td>
<td>Nominalization, Predication and Type Containment, p. 40.</td>
</tr>
<tr>
<td>92/24</td>
<td>M. Codish, D. Dams, Eyal Yardeni</td>
<td>Bottom-up Abstract Interpretation of Logic Programs, p. 33.</td>
</tr>
<tr>
<td>92/25</td>
<td>E. Poll</td>
<td>A Programming Logic for Fω, p. 15.</td>
</tr>
</tbody>
</table>
| 93/01 | R. van Geldrop                 | Deriving the Aho-Corasick algorithms: a case study into the
|                                 | synergy of programming methods, p. 36.                     |
| 93/02 | T. Verhoeff                    | A continuous version of the Prisoner's Dilemma, p. 17      |
| 93/03 | T. Verhoeff                    | Quicksort for linked lists, p. 8.                          |
| 93/04 | E. H. L. Aarts, J. H. M. Korst, P. J. Zwietering | Deterministic and randomized local search, p. 78.         |