Memetic Algorithms with Variable-Depth Search to Overcome Local Optima

Dirk Sudholt*
Fakultät für Informatik, LS 2
Technische Universität Dortmund
Dortmund, Germany

ABSTRACT

Variable-depth search (shortly VDS) is well-known as Lin-Kernighan strategy for the TSP and Kernighan-Lin for graph partitioning. The basic idea is to make a sequence of local moves and to freeze all moved combinatorial objects to prevent the search from looping. VDS stops when no further local move is possible and returns a best found solution.

We analyze memetic algorithms with VDS for three binary combinatorial problems: Mincut, Knapsack, and Maxsat. More precisely, we focus on simply structured problem instances containing local optima that are very hard to overcome. Many common trajectory-based algorithms fail to find a global optimum: the (1+1) EA, iterated local search, and simulated annealing need exponential time with high probability. However, memetic algorithms using VDS easily manage to find a global optimum in expected polynomial time. These results strengthen the usefulness of memetic algorithms with VDS from a theoretical perspective.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Theory, Algorithms, Performance

Keywords

Memetic algorithms, runtime analysis, combinatorial optimization, Kernighan-Lin, variable-depth search, simulated annealing

*Supported by the Deutsche Forschungsgemeinschaft (DFG) as a part of the Collaborative Research Center "Computational Intelligence" (SFB 531).

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

1. INTRODUCTION

Over the past decades, a plethora of randomized search heuristics has been proposed, analyzed, and applied to problems from combinatorial optimization. Simulated annealing is a simple hill-climber that may, however, accept worse solutions with a probability monotone in the current *temperature*. The temperature is typically decreased over time, which gradually changes the focus from exploration to exploitation. Evolutionary algorithms create new solutions by mutation and/or recombination and try to obtain good solutions by selecting individuals with high fitness. Memetic algorithms [6] additionally incorporate local search into this search process to speed up the evolution. One such algorithm is known as iterated local search [13] where local search is used in every generation to find a local optimum. Then a large *kick move* or *perturbation* is performed in the hope for a basin of attraction of a better local optimum.

1.1 Strategies to Overcome Local Optima

All these heuristics have to deal with the possibility of reaching a poor local optimum. Population-based heuristics usually rely on diversifying the search to explore different local optima. However, it may happen that the whole population converges to non-global local optima and then the situation is not too different from trajectory-based algorithms, that is, algorithms maintaining only a single current solution. These algorithms have to rely on different mechanisms to overcome local optima.

We describe some common approaches to overcome local optima and give pointers to related theoretical works, with a focus on combinatorial optimization.

1. **Accept solutions with inferior fitness:**
   Simulated Annealing may accept solutions with inferior fitness, enabling the algorithm to climb down a hill. This well-known strategy proved to be effective for graph bisection (Jerrum and Sorkin [10]), the two-dimensional Ising model (Fischer [3]), and minimum spanning trees (Wegener [22]).

2. **Decrease the attractiveness of local optima:**
   Tabu search maintains a *tabu list* of solutions that are "taboo" for the algorithm, hence making the local optimum less attractive. Diversity mechanisms like fitness sharing may as well decrease the attractiveness of a local optimum. If many individuals are concentrated on one local optimum, they are forced to "share" their fitness according to their similarity. Hence, the algorithm is encouraged to decrease similarity in the population.

The effectiveness of fitness sharing has been shown recently (Friedrich, Oliveto, Sudholt, and Witt [4]) for a toy problem that, however, has the same structure as the Mincut instance we will investigate in Section 4.

3. Use larger or multiple neighborhoods:
Evolutionary algorithms like the well-known (1+1) EA use a stochastic neighborhood where every search point has a positive probability to be created. This simple property enables the approximation of maximum matchings (Giel and Wegener [5]) and balanced partitions (Witt [23]). In addition, evolutionary algorithms may use crossover to recombine different local optima. The usefulness of crossover was first shown for toy problems (Jansen and Wegener [8]). Moreover, crossover together with fitness sharing is effective for (a problem equivalent to) 2-coloring binary trees (Sudholt [18]). The idea of using different neighborhoods is dominant in variable neighborhood search algorithms (see, e. g., [14]) exploiting that a local optimum w. r. t. one neighborhood need not be a local optimum w. r. t. another one. Memetic algorithms like iterated local search also fall into this category as they combine mutation and local search. Their usefulness so far has only been proven for artificial problems (Sudholt [20]).

Another common strategy is to restart the algorithm after convergence to local optima. This can be seen as a very large perturbation and hence falls into the third category.

1.2 Variable-depth search

In this work we will consider a special local search operator for use within the framework of iterated local search, so-called *variable depth search* (VDS). Variable-depth search is well-known for the TSP as *Lin-Kernighan* strategy [12] and for Maxcut as *Kernighan-Lin* [11]. The idea is to perform a sequence of local moves. The next local move is chosen in a greedy fashion. If there is a local move that increases fitness, then a move with maximal fitness gain is chosen. Otherwise, a move with minimal loss in fitness is selected. To prevent the algorithm from looping, certain parts of the search space are made "tabu". For binary search spaces this means that if VDS flips some bit, then this bit cannot be flipped again during the run of VDS. The output of VDS is then a best solution from the sequence of local moves.

Iterated local search using VDS combines approaches from all three mentioned categories to overcome local optima. Firstly, it easily traverses solutions with inferior fitness if no fitness-improving move is available. Secondly, starting with a local optimum this local optimum is made "tabu" like in tabu search since steps moving back towards the local optimum are not allowed. The only difference to classical tabu search is that we do not keep a tabu list of single individuals, but render large parts of the search space tabu. Finally, we employ different neighborhoods: VDS and mutation as perturbation.

This combination of strategies makes it easy for VDS to overcome local optima. Another remarkable aspect is the greedy component in VDS since we always choose a best move among the feasible local moves. We will in the following give examples where this greedy behavior provides a good guidance for the algorithm in order to find a global optimum.

1.3 Our Results

We will investigate memetic algorithms with VDS on combinatorial problems using rigorous runtime analyses. Such rigorous results for memetic algorithms are scarce; the only rigorous runtime analyses of memetic algorithms appeared only recently (Sudholt [19, 20]). The investigated problems are artificial pseudo-Boolean functions, defined to demonstrate the impact of the parametrization on the performance of memetic algorithms. Our goal is to extend such analyses to problems from combinatorial optimization.

We investigate instances of problems from combinatorial optimization, namely Mincut, Knapsack, and Maxsat. Definitions and descriptions of these problems are postponed to the following sections. The chosen instances contain non-global local optima with large basins of attraction that are hard to overcome. We will see that memetic algorithms with VDS are efficient on these functions while common trajectory-based algorithms like the (1+1) EA, traditional iterated local search algorithms, and simulated annealing fail to find a global optimum, even if they are given exponential time.

The instances we consider have a very simple structure. This helps to keep the argumentation simple and to focus on the essentials. We assume that the reader is familiar with basic knowledge on common combinatorial optimization problems. For details, we refer to appropriate text books, e. g. [16, 1, 21].

The remainder of this paper is structured as follows. First, we define all investigated algorithms in Section 2. Section 3 contains lower bounds on the runtime after the population has converged to local optima. In Sections 4, 5, and 6 we then deal with instances for the problems Mincut, Knapsack, and Maxsat, respectively. We conclude in Section 7.

2. ALGORITHMS

The most natural metric in the search space $\{0,1\}^n$ is the Hamming distance $\mathrm{H}(x,y)$ between x and y, i. e., the number of bits differing in these two search points. With $\mathcal{N}_1(x) = \{y \mid \mathrm{H}(x,y) = 1\}$ we denote the open Hamming neighborhood of x. In general, $\mathcal{N}_d(x)$ contains all y with Hamming distance exactly d to x.

These notions are extended to a set $S \subseteq \{0,1\}^n$ as follows. $\mathrm{H}(x,S) = \min_{y\in S}\{\mathrm{H}(x,y)\}$ denotes the Hamming distance from x to S. Similarly, $\mathcal{N}_d(S) = \{y \mid \mathrm{H}(y,S) = d\}$ contains all points with Hamming distance d to S. The *diameter* of a neighborhood $\mathcal{N}$ is defined as the largest Hamming distance between any $x, y \in \mathcal{N}(z)$. We restrict ourselves to symmetric neighborhoods in a sense that $\mathrm{H}(x,z) = \mathrm{H}(y,z)$ implies either $x, y \in \mathcal{N}(z)$ or $x, y \notin \mathcal{N}(z)$.

We first define two local search operators used throughout this work. Standard local search accepts any neighbor with strictly larger fitness and stops whenever a local optimum is reached or the number of iterations exceeds a predefined value $\delta(n)$ called *local search depth*. The pivoting rule is random, here, although results can easily be adapted to other pivoting rules like always choosing the best neighbor.

OPERATOR 1 (STANDARD LOCAL SEARCH).
For $\delta(n)$ iterations do
  Choose a random $z \in \mathcal{N}(y)$ with $f(z) > f(y)$
    or stop with output y if no such z exists.
  Let $y := z$.
Return y.

We already explained the concept of VDS. In the following procedure Z denotes the sequence of solutions encountered during VDS and L is a set of indices for all bits that have been locked.

OPERATOR 2 (VARIABLE-DEPTH SEARCH).
$Z, L := \emptyset$.
While $V_y := \{z \in \mathcal{N}(y) \mid \forall i\colon (y_i \neq z_i \Rightarrow i \notin L)\} \neq \emptyset$ {
  Choose a random $z \in V_y$ with maximal f-value.
  $Z := Z \cup \{z\}$.
  $L := L \cup \{i \mid y_i \neq z_i\}$.
  $y := z$.
}
Return a random $z \in Z$ with maximal f-value.

Note that one VDS takes at most n iterations since in every iteration of the loop at least one index is added to L (we assume $y \notin \mathcal{N}(y)$). Using one of these local search operators, we now define a memetic algorithm with population size 1, the (1+1) Memetic Algorithm or shortly (1+1) MA. Mutation is done by flipping each bit independently with a fixed *mutation probability* p_m.

ALGORITHM 1 ((1+1) MEMETIC ALGORITHM).
Choose x uniformly at random.
Repeat

Mutation:	Create y by flipping each bit in x with probability p_m.
Local Search:	Decide whether to use local search. If "yes" then $y :=$ local search(y).
Selection:	If $f(y) \geq f(x)$ then $x := y$.

The (1+1) MA never using local search equals the evolutionary algorithm called (1+1) EA (see, e. g., [5, 9, 23]). Alternatively, local search may be applied with a fixed frequency [19]. We may also choose to apply local search probabilistically with a fixed probability as done in [7]. The algorithm where local search is called in each generation is called *iterated local search* [13]. In particular, we will often refer to *iterated VDS* as the (1+1) MA calling VDS in every generation.

All algorithms considered so far have in common that the best-so-far fitness cannot decrease. Simulated annealing always accepts better solutions, but it also allows worse solutions to be accepted. This decision is made dependent on the size of the fitness decrease and a parameter called the *temperature*. If the temperature equals 0, simulated annealing behaves like a hill-climber, i. e., it doesn't accept worsenings. The larger the temperature, the more likely it is to accept worse solutions. It is common practice to start with a high temperature and then to decrease the temperature over time. This way, simulated annealing can explore the search space in the beginning and then gradually turns into a hill-climber focusing on exploitation. A strategy to turn down the temperature is called a *cooling schedule*. Unless otherwise noted, simulated annealing uses the Hamming neighborhood $\mathcal{N} = \mathcal{N}_1$. It is formulated for maximization to match previously defined algorithms.

ALGORITHM 2 (SIMULATED ANNEALING).
Choose x uniformly at random.
Let $t := 0$.
Repeat
  Choose $y \in \mathcal{N}(x)$ uniformly at random.
  Set $x := y$ with prob. $\min\{1, \exp((f(y) - f(x))/T(t))\}$.
  $t := t + 1$.

Simulated annealing with a fixed temperature is called *Metropolis algorithm*. It was long unknown whether cooling down the temperature is essential for natural problems, i. e., whether simulated annealing outperforms the Metropolis algorithm with an optimal temperature. This question was recently solved in the affirmative by Wegener [22] for the natural problem of computing a minimum spanning tree.

For the efficiency of an algorithm a plausible performance measure is the number of generations until a global optimum is found. We will also consider the number of function evaluations, referred to as *optimization time*. Thereby, we in particular account for the computational effort of local search. Note that the number of function evaluations in a generation with local search is bounded above by $\delta(n) \cdot |\mathcal{N}|$ for greedy local search and by $n \cdot |\mathcal{N}|$ for VDS.

3. LOWER BOUNDS WHEN STUCK IN LOCAL OPTIMA

We start our investigations with lower bounds for different algorithms after convergence to local optima. A local optimum is difficult for an algorithm if it has a large basin of attraction. This is especially true if the fitness decreases with any local move leading away from the local optimum. Combinatorial fitness landscapes often contain several local optima that are close to one another. Therefore, it makes sense to group such local optima into sets.

Definition 1. A non-empty set $S^* \subseteq \{0,1\}^n$ is called α-difficult for $\alpha = \alpha(n)$ w. r. t. the function f and a neighborhood $\mathcal{N}$ if $z \in \mathcal{N}(y)$ and $\mathrm{H}(y, S^*) < \mathrm{H}(z, S^*) \leq \alpha$ implies $f(y) > f(z)$ for any $y, z \in \{0,1\}^n$ and S^* does not contain global optima.

The definition of α-difficulty implies that all search points with Hamming distance less than α to S^* have worse fitness than any point in S^*. This immediately leads to lower bounds for the (1+1) MA once S^* has been reached.

LEMMA 1. *Let S^* be α-difficult. If the (1+1) MA using standard local search with neighborhood of diameter $d = O(1)$ and mutation probability $p_m \leq (1-\varepsilon) \cdot \alpha/n$ for some $\varepsilon > 0$ reaches S^*, the remaining time until a global optimum is found is $2^{\Omega(\alpha)}$ with probability at least $1 - 2^{-\Omega(\alpha)}$.*

PROOF. W. l. o. g. α grows with n, otherwise the theorem is trivial. Apart from individuals in S^*, the (1+1) MA only accepts an offspring if mutation creates a solution with Hamming distance at least $\alpha - d$ to its parent as otherwise local search runs back into S^* or the generation ends with an inferior solution. With mutation probability at most $(1-\varepsilon) \cdot \alpha/n$ the expected number of flipping bits in one mutation is at most $(1-\varepsilon) \cdot \alpha$. The probability that at least $\alpha - d$ bits flip is at most $2^{-\Omega(\alpha)}$ by Chernoff bounds (see, e. g., [15]). The probability that this happens at least once in $2^{c\alpha}$ steps, $c > 0$ a small enough constant, is still of order $2^{-\Omega(\alpha)}$. □

Simulated annealing can accept worse solutions with a certain probability that depends on the loss in fitness and the current temperature $T = T(t)$. Escaping a single local optimum is easy if the temperature is high enough such that all local moves have a good chance to be accepted. The reason is simple: if the current solution is close to the local optimum, there are more local moves leading away from it than moving closer to it. If the temperature is too low (or

has been cooled down too fast), escaping a local optimum is much more difficult. We can make this precise for a scenario where the temperature leaves us with a noticeable bias towards search points with high fitness.

LEMMA 2. *Let S^* be α-difficult and let $|f(x) - f(y)| \geq \Delta = \Delta(\alpha)$ if $y \in \mathcal{N}(x)$ and x, y have Hamming distance at most α to S^*. If simulated annealing with temperature $T(t) \leq \Delta/(\ln(4n/\alpha))$ reaches a search point with Hamming distance at most $\alpha/2$ to S^*, the remaining optimization time is $2^{\Omega(\alpha)}$ with probability $1 - 2^{-\Omega(\alpha)}$.*

PROOF. Observe $\mathrm{H}(y, x^*) \geq \mathrm{H}(y, S^*)$ for any y and any $x^* \in S^*$. This allows us to focus on a single search point $x^* \in S^*$ with minimal Hamming distance to the current search point x. If $\mathrm{H}(x, S^*) = k$ and $\alpha/2 \leq k < \alpha$, the probability to increase the Hamming distance to x^* (and hence S^*) by 1 is

$$p^+ \;\leq\; \frac{n-k}{n} \cdot e^{-\Delta/T} \;<\; e^{-\Delta/T} \;\leq\; \frac{\alpha}{4n}$$

due to the assumption on T. On the other hand, the probability to decrease the Hamming distance to S^* by 1 is

$$p^- \;\geq\; \frac{k}{n} \;\geq\; \frac{\alpha}{2n}$$

as all k steps moving closer to x^* are accepted. Hence, conditional on an exchange of the current search point, the probability to move away from S^* is at most 1/3.

Consider the first point of time where simulated annealing creates a search point x with $\mathrm{H}(x, S^*) = \alpha/2 + 1$ (w. l. o. g. assuming α to be even). We now argue that with high probability simulated annealing returns to distance $\alpha/2$ before reaching distance α. By gambler's ruin arguments [15], this probability is at least

$$\frac{2^{\alpha/2} - 2}{2^{\alpha/2} - 1} \;=\; 1 - \frac{1}{2^{\alpha/2} - 1} \;=\; 1 - 2^{-\Omega(\alpha)}.$$

If we create a search point with distance at most $\alpha/2$, we wait until the next solution with distance $\alpha/2 + 1$ is created and repeat the argumentation. The time for one such trial is trivially bounded below by 1. By the union bound, the probability that $2^{c\alpha}$ trials fail to create a solution with distance α is still of order $2^{-\Omega(\alpha)}$ if $c > 0$ is small enough. □

An α-difficult local optimum is challenging for memetic algorithms and simulated annealing as both have difficulties with large "valleys" in the fitness landscape. This similarity between evolutionary algorithms and simulated annealing has already been recognized by Jansen and Wegener [9].

4. MINCUT

Given an undirected graph $G = (V, E)$, the problem Mincut is to partition all vertices into two non-empty subsets V_0, V_1 such that the number of edges between V_0 and V_1 is minimized. Such edges are called *cut edges*. We remark that specialized algorithms can solve the Mincut problem in polynomial time, even in the case of weighted graphs [17].

Given an ordering of the vertices $V = \{v_1, \ldots, v_n\}$, we obtain a binary representation $x = x_1 \ldots x_n \in \{0,1\}^n$ for $n = |V|$ such that $v_i \in V_{x_i}$ for every i. If V_0 and V_1 are non-empty, the fitness function is encoded as follows. The fitness is chosen as the number of non-cut edges, written as $\sum_{\{u,v\} \in E}(x_u x_v + (1 - x_u)(1 - x_v))$. However, if V_0 or V_1 is empty, i. e., $x \in \{0^n, 1^n\}$, the non-emptiness constraint is violated and we penalize such a solution by assigning a negative f-value.

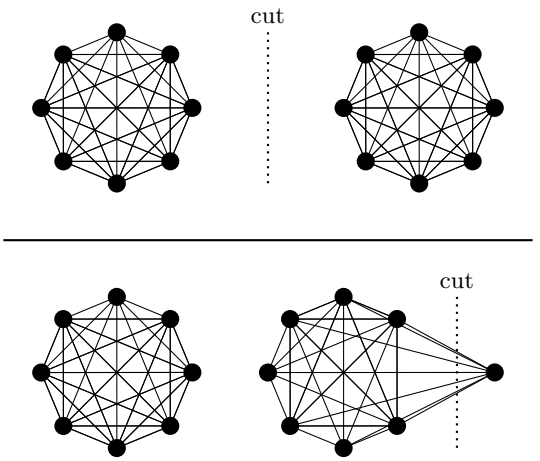

Figure 1: A global optimum (top) and a local optimum (bottom) for the Mincut instance with $n = 16$.

$$f(x) := \begin{cases} \sum_{\{u,v\} \in E} (x_u x_v + (1 - x_u)(1 - x_v)) & \text{if } x \notin \{0^n, 1^n\} \\ -1 & \text{if } x \in \{0^n, 1^n\} \end{cases}$$

Consider the following instance $G = (V, E)$ that consists of two cliques of size $n/2$, each (see Figure 1).

$$V = \{u_1, \ldots, u_{n/2}, v_1, \ldots, v_{n/2}\},$$
$$E = \{\{u_i, u_j\}, \{v_i, v_j\} \mid 1 \leq i < j \leq n/2\}.$$

Consider a partition $V = V_0 \cup V_1$ where w. l. o. g. $u_1 \in V_0$. Obviously, the optimal partition is $V_0 = \{u_1, \ldots, u_{n/2}\}$ with a cut of size 0. All partitions with $|V_0| = 1$ or $|V_0| = n - 1$ are locally optimal with a cut size of $n/2$ as V_0 and V_1 are constrained to be non-empty (see Figure 1). These points form a set S^* that is α-difficult for $\alpha = \lfloor n/4 - 1 \rfloor$ w. r. t. $\mathcal{N}_1$ as every local move shifting a vertex to the smaller set of the partition increases the cut size, unless it contains at least half of a clique.

THEOREM 1. *Consider the (1+1) MA with standard local search using neighborhood $\mathcal{N}_1$ and mutation probability $p_m \leq 1/5$. The optimization time for the Mincut instance is at least $2^{\Omega(n)}$ with probability $1/2 - O(1/\sqrt{n})$.*

PROOF. We already argued that all local optima are α-difficult for $\alpha = \lfloor n/4 - 1 \rfloor$. If n is large enough, we have $1/5 \leq (1 - \varepsilon) \cdot \alpha/n$ for an appropriate $\varepsilon > 0$, hence the claim follows from Lemma 1 if we can prove that the algorithm reaches S^* with probability $1/2 - O(1/\sqrt{n})$.

Let S_0 contain all four (feasible and infeasible) solutions with no cut edge. Let $S_1 = \mathcal{N}_1(S_0)$. We now argue that with high probability the algorithm evaluates a solution in S_1 before evaluating one from S_0. Consider the first generation where a solution in $S_0 \cup S_1$ is created. Let p_0 and p_1 denote the probabilities that this solution belongs to S_0 and S_1, respectively. We claim that $p_1 = \Omega(p_0 \cdot \sqrt{n})$. This implies that S_1 is found before S_0 with probability $1 - O(1/\sqrt{n})$.

Initialization creates a search point in S_0 with probability 2^{-n+2}. Assume that the algorithm doesn't start in $S_0 \cup S_1$.

Consider a generation with current solution $x \notin (S_0 \cup S_1)$ and the next offspring creation. If $S_0 \cup S_1$ is reached during local search, the claim is trivial. Hence we focus on mutation only and fix a search point $z \in S_0$. If x has Hamming distance k to z, we have k solutions in S_1 with Hamming distance $k-1$ to x and $n-k$ solutions in S_1 with Hamming distance $k+1$. The probability of reaching a specific $y \in \mathcal{N}_1(z)$ differs from the probability $\mathrm{P}(z)$ to reach z in just one bit position. More precisely, the probabilities differ by factors $p_m/(1-p_m)$ or $(1-p_m)/p_m$, dependent on whether this bit has to be flipped or not. Let $\mathrm{P}(\mathcal{N}_1(z))$ denote the probabiliy to reach $\mathcal{N}_1(z)$, then

$$\begin{aligned}\mathrm{P}(\mathcal{N}_1(z)) &\geq k \cdot \mathrm{P}(z) \cdot \frac{1-p_m}{p_m} + (n-k) \cdot \mathrm{P}(z) \cdot \frac{p_m}{1-p_m} \\ &= \mathrm{P}(z) \left(\frac{k(1-p_m)^2 + (n-k)p_m^2}{p_m(1-p_m)} \right) \\ &= \mathrm{P}(z) \left(\frac{k(1-2p_m) + np_m^2}{p_m(1-p_m)} \right)\end{aligned}$$

We see that this term is increasing with k, hence in the worst case $k = 2$. Along with $p_m \leq 1/5$, we arrive at the bound

$$\mathrm{P}(\mathcal{N}_1(z)) \geq \mathrm{P}(z) \left(\frac{1+np_m^2}{p_m} \right).$$

In case $p_m \leq 1/\sqrt{n}$ the term in brackets is at least $1/p_m \geq \sqrt{n}$. If $p_m > 1/\sqrt{n}$, then this term is at least $np_m \geq \sqrt{n}$ as well. We conclude

$$\mathrm{P}(\mathcal{N}_1(z)) \geq \mathrm{P}(z) \cdot \sqrt{n}.$$

Since this holds for all $z \in S_0$, $p_1 \geq p_0 \cdot \sqrt{n}/2$ follows and S_1 is found before S_0 with probability $1 - O(1/\sqrt{n})$.

As long as no local optimum is found, the fitness is indifferent to the question whether the majority of the u-vertices is in V_0 or in V_1. The same holds independently for the v-vertices. Hence, if the first cut in S_1 is created, given that S_0 has not been found yet, all cuts in S_1 have the same probability to be found. Half of these cuts are local optima, hence the probability that a local optimum is found equals $1/2$. By the union bound, the probability to reach a local optimum is at least $1/2 - O(1/\sqrt{n}) - 2^{-n+2} = 1/2 - O(1/\sqrt{n})$. □

Reusing ideas from the proof of Theorem 1, we show that also simulated annealing fails with probability close to 1/2.

THEOREM 2. *Simulated annealing with an arbitrary cooling schedule where $T(t)$ is monotone decreasing needs at least $2^{\Omega(n)}$ steps for the Mincut instance with probability $1/2 - 2^{-\Omega(n)}$.*

PROOF. We divide a run into two phases: the first phase ends when the temperature first drops to $n/12$ and then the second phase starts. Let T_1 be the number of generations in Phase 1 and let S_0 and S_1 be defined as in the proof of Theorem 1. We first prove that in Phase 1 no solution in S_0 will be evaluated in exponential time, with high probability.

Consider a search point x with $\mathrm{H}(x, S_0) = k \leq n/(4e^6)$. The probability to increase the Hamming distance to S_0 is

$$p^+ \geq \frac{n-k}{n} \cdot e^{-n/(2T)} \geq \frac{1}{2} \cdot e^{-6}$$

as the worst fitness decrease equals $n/2$. The probability to decrease the Hamming distance to S_0 equals

$$p^- = \frac{k}{n} \leq \frac{1}{4} \cdot e^{-6}.$$

Together, the conditional probability to decrease the Hamming distance is bounded by 1/3, provided that the Hamming distance is changed.

The probability of initializing simulated annealing with a search point x such that $\mathrm{H}(x, S_0) \leq n/(4e^6)$ is $2^{-\Omega(n)}$. Repeating the gambler's ruin arguments from the proof of Lemma 2, the probability that S_0 is reached before a point with Hamming distance larger than $n/(4e^6)$ to S_0 is reached is $2^{-\Omega(n)}$. The probability that this happens within the first $\min\{T_1, 2^{cn}\}$ steps is still of the same order if $c > 0$ is small enough.

This concludes the proof if $T_1 \geq 2^{cn}$. Otherwise, we consider Phase 2 and assume that S_0 has not been reached in Phase 1. By the locality of the search operator S_1 is reached before S_0 and the probability that a local optimum is found equals 1/2.

Given a fixed Hamming distance $k \leq n/12$ from S_0, the minimal fitness difference Δ between two neighbors, as defined in Lemma 2, is attained when only one clique is cut. In that case the cut clique has k vertices on one side of the partition and when moving a $(k+1)$-st vertex, the fitness decreases by $n/2 - 2k \geq n/3$. Applying Lemma 2 with $\alpha = n/12$ and $\Delta(\alpha) = n/3$ proves the claim for $T \leq \Delta/(\ln(4n/\alpha)) = 1/(3\ln(48))$ and hence for $T \leq n/12$. □

We have seen that the local optima of the Mincut instance are extremely hard for standard evolutionary algorithms, memetic algorithms, and simulated annealing. In contrast to this, iterated VDS easily escapes from this local optimum. The following proof is surprisingly simple.

THEOREM 3. *Iterated VDS without mutation optimizes the Mincut instance in at most 2 generations with probability 1.*

PROOF. The first VDS reaches a global or local optimum. Assume that a local optimum is reached and that w. l. o. g. V_0 consists of a single u-vertex. In the next call, VDS starts moving u-vertices to V_0 as these operations lead to a minimal fitness decrease. If at least half of the u-clique is contained in V_0, the fitness even increases when adding more u-vertices. Once all u-vertices have been moved to V_0, a global optimum is found. □

5. KNAPSACK

The Knapsack problem is a well-known NP-hard combinatorial problem. Suppose we are given a knapsack that can hold objects up to a specified weight limit G. Among a set of objects with associated profit values $v_1, \ldots, v_n$ and weights $g_1, \ldots, g_n$, we have to select objects for the knapsack such that the total profit is maximized while respecting the weight limit.

As a fitness function, we take the profit of all chosen objects if the weight limit is respected. Otherwise, the fitness function gives hints to drop selected objects.

$$f(x) := \begin{cases} \sum_{i=1}^n x_i v_i & \text{if } \sum_{i=1}^n x_i g_i \leq G \\ -\sum_{i=1}^n x_i & \text{if } \sum_{i=1}^n x_i g_i > G \end{cases}$$

Consider the following Knapsack instance I for odd n and $N = (n+1)/2$.

$$\begin{aligned} 1 \leq i \leq N &: v_i = g_i = n \\ N < i \leq n &: v_i = g_i = n+1 \\ G &= N \cdot n \end{aligned}$$

For all objects profit equals weight. Hence, this instance also represents an instance of the subset sum problem, a restricted formulation of Knapsack.

Call the objects with weight $n+1$ *big* and the other ones *small*. The weight limit is chosen such that all N small objects exactly fit into the knapsack. This selection yields a total profit of $(n^2+n)/2$. On the other hand, if one big object is chosen, there is only space for a total number of $N-1$ objects. If the current packing also contains at least one small object, the profit may be increased by dropping a small object and adding a big object, which increases the profit by 1. Thus, a packing of all $N-1$ big objects is locally optimal with a profit of $(N-1)\cdot(n+1) = (n^2-1)/2$.

We see that we have a non-optimal local optimum with all big objects and a unique global optimum with N small objects. Furthermore, when reaching the local optimum exchanging a big object for a small one decreases the fitness. Only after all big objects have been exchanged for small ones, an N-th small object may be added to yield a global optimum.

THEOREM 4. *Consider the (1+1) MA with standard local search using any neighborhood with diameter $d = O(1)$ and mutation probability $p_m \leq 1/2$. The optimization time for the Knapsack instance is $2^{\Omega(n)}$ with probability $1 - 2^{-\Omega(n)}$. The same holds for simulated annealing with constant-diameter neighborhood and an arbitrary cooling schedule.*

PROOF. Let S_k be the set of packings with k selected objects. Let $A <_f B$ for $A, B \subseteq \{0,1\}^n$ if all search points in A have lower fitness than all search points in B. For the instance I then

$$S_0 <_f S_1 <_f \cdots <_f S_{N-1} \text{ and}$$
$$S_n <_f S_{n-1} <_f \cdots <_f S_{N+1} <_f (S_N \setminus \mathrm{OPT})$$

where OPT denotes the unique global optimum $1^N 0^{N-1}$. This also holds for a modified instance I^* containing n objects with profit and weight $n+1$ and weight limit $G^* = G$. As long as OPT has not been found, both the (1+1) MA and simulated annealing behave similarly on I and I^*. Only when choosing between two packings within S_k a different behavior may occur: on I a solution with more big objects will be preferred, while the situation is completely symmetric on I^*. By the assumption $p_m \leq 1/2$ a tendency towards big objects cannot help to find solutions with many small objects. We will in the following estimate the probability of getting close to the global optimum, that is, to have significantly more small objects than big ones. Therefore, we are pessimistic when considering I^* instead of I.

The global optimum OPT for I can only be found if mutation and/or local search create a search point in $Z = \{x \mid \mathrm{H}(x, \mathrm{OPT}) \leq d\}$. Let x_t be the t-th evaluated search point. The probability that during the first T evaluations no search point in Z is found is at most $\sum_{t=1}^{T} \mathrm{P}(x_t \in Z)$. Fix t, then

$$\mathrm{P}(x_t \in Z) = \sum_{k=0}^{n} \mathrm{P}(x_t \in Z \mid x_t \in S_k) \cdot \mathrm{P}(x_t \in S_k).$$

Observe that due to the perfect symmetry of S_k, each point in S_k is equally likely to be x_t. Moreover, the size of Z is polynomially bounded while S_k has size $2^{\Omega(n)}$ for $k = n/2 \pm O(1)$. Hence, $\mathrm{P}(x_t \in Z \mid x_t \in S_k) = 2^{-\Omega(n)}$ and the probability to find Z within the first T generations is at most

$$\sum_{t=1}^{T} \mathrm{P}(x_t \in Z) \leq \sum_{t=1}^{T} \sum_{k=0}^{n} 2^{-\Omega(n)} \cdot \mathrm{P}(x_t \in S_k) = T \cdot 2^{-\Omega(n)}.$$

Choosing $T = 2^{cn}$ for a small enough positive constant c proves that on instance I^* the algorithm doesn't create OPT or a point in its neighborhood in exponential time, with overwhelming probability. As chances to reach the optimum on I are not better than for I^*, the theorem follows. □

THEOREM 5. *Iterated VDS using neighborhood $\mathcal{N}_1 \cup \mathcal{N}_2$ without mutation optimizes the Knapsack instance within 2 generations with probability 1.*

PROOF. After initialization, VDS either runs into the local or the global optimum. Suppose that we have found the local optimum of $N-1$ big objects and consider the next call of VDS. The least decrease in fitness is to exchange a big object for a small one. This is repeated until all big objects have been replaced by small ones and then the last small object is added. □

6. MAXSAT

Maxsat is another well-known and important combinatorial problem. Given n Boolean variables $x_1, \ldots, x_n$ a literal is either a variable or a negated variable. A clause is a disjunction of literals; for example $(x_1 \vee \overline{x_3} \vee x_4)$ is a clause with three literals. We say that a clause is satisfied w. r. t. an assignment x to the variables if the clause evaluates to true. Given a set C of clauses, the problem Maxsat asks for an assignment of the variables such that the number of satisfied clauses is maximized. This problem is known to be NP-hard even if all clauses only contain 2 literals.

A natural choice of the fitness function is to choose the number of satisfied clauses. This function has already been investigated by Droste, Jansen, and Wegener [2] on the following instance.

$$\forall i \neq j \neq k \neq i\colon (x_i \vee \overline{x_j} \vee \overline{x_k}) \in C$$
$$(x_1), (x_2), \ldots, (x_n) \in C$$

An important observation is that this instance is symmetric in a sense that all variables are treated equally. Note that every clause has exactly one non-negated literal, hence the assignment 1^n satisfies every clause. On the other hand, most clauses contain two negated literals. This gives strong hints for a search heuristic to set variables to 0. Due to this deceptive property Papadimitriou [16] first defined this instance as a worst-case example for the performance of a heuristic algorithm for Maxsat.

Due to symmetry of the instance, we can formulate the fitness as a function of unitation, i. e., $f(x)$ only depends on the number of 1-bits in x, denoted by $|x|_1$. If $|x|_1 = i$, then i unit clauses (i. e. clauses with just one literal) are satisfied. Among the other $\binom{n}{3}$ clauses there are $n \cdot \binom{i}{2}$ clauses where the last two literals evaluate to false. Moreover, there are $(n-i)$ choices for the first variable such that the first literal also evaluates to false. Hence, $(n-i)\cdot\binom{i}{2}$ clauses of length 3 are unsatisfied. We conclude that the number of satisfied clauses and hence the fitness is given by the formula

$$f(x) = \binom{n}{3} - (n - |x|_1) \cdot \binom{|x|_1}{2} + |x|_1.$$

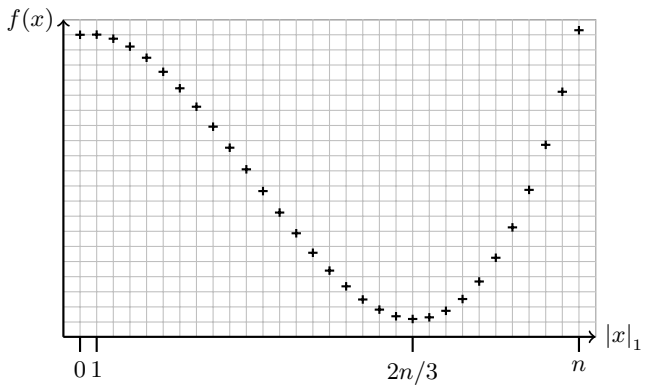

Figure 2: Sketch of the fitness landscape according to the Maxsat instance with $n = 30$.

It is easy to see that $|x|_1 = n$ implies a global optimum with fitness $\binom{n}{3} + n$. The search point 0^n has fitness $\binom{n}{3}$ and all x with $|x|_1 = 1$ have fitness $\binom{n}{3} + 1$ as $\binom{1}{2} = 0$. Assuming $n \geq 6$ and n multiple of 3, the fitness decreases with $|x|_1$ in the interval $[1, 2n/3]$. Thus, $S^* = \{x \mid |x|_1 = 1\}$ is $2n/3$-difficult for the neighborhood $\mathcal{N}_1$. A sketch of the function f is shown in Figure 2.

Droste, Jansen, and Wegener [2] proved for a very large class of evolutionary algorithms with mutation probability $p_m \leq 1/2$ that these algorithms need exponential time with overwhelming probability. A look at their proof reveals that they even show a stronger result. The borderline between the basin of attraction of all local optima and the globally optimal one is located around search points with $2n/3$ 1-bits. Droste, Jansen, and Wegener even prove that the considered algorithms fail in creating a search point with at least $(1/2 + \varepsilon)n$ 1-bits for an arbitrary small positive constant ε. Choosing $\varepsilon < 1/6$ we can safely conclude that even if standard local search with constant-diameter neighborhood is applied after mutation no search point with at least $2n/3$ 1-bits is found.

We give a self-contained formulation of their result, describing all necessary properties of the investigated class of algorithms.

THEOREM 6 (DROSTE, JANSEN, AND WEGENER [2]). *Let A be an evolutionary algorithm with a population of at most polynomial size initialized uniformly at random. Allow A to perform an arbitrary combination of two types of operators: mutation with mutation probability $p_m \leq 1/2$ and selection. The only requirement to selection operators is that with $f(x) \geq f(y)$ the probability to select x is not less than the probability to select y.*

Then the probability that during the first $2^{o(n^{1/2})}$ offspring creations A finds an individual with at least $(1/2+\varepsilon)n$ 1-bits, $\varepsilon > 0$, is bounded by $2^{-\Omega(n^{1/2})}$.

We see that this claim covers the (1+1) EA with mutation probability $p_m \leq 1/2$ as well as simulated annealing with an arbitrary cooling schedule. Choosing, say, $\varepsilon = 1/8$ yields that the algorithm with overwhelming probability even stays at a distance $\Omega(n)$ to all search points with at least $2n/3$ 1-bits. Hence, even if standard local search with constant-diameter neighborhood is applied at any time, the algorithm cannot create a search point with more than $(5/8)n$ 1-bits. This argument not only holds for one step. Instead, Theorem 6 remains valid if we allow standard local search as additional operator. Exploiting $t \cdot 2^{-\Omega(n)} = 2^{-\Omega(n)}$ for $t = 2^{cn}$, c a small enough positive constant, yields the following theorem.

THEOREM 7. *Consider the (1+1) MA with standard local search using any neighborhood with diameter $d = O(1)$ and mutation probability $p_m \leq 1/2$. The optimization time for the Maxsat instance is at least $2^{\Omega(n)}$ with probability $1 - 2^{-\Omega(n)}$. The same also holds for simulated annealing with an arbitrary cooling schedule.*

Again, we ask ourselves what iterated VDS can do. Interestingly, iterated VDS without mutation is not effective for the Maxsat instance.

THEOREM 8. *Iterated VDS with neighborhood $\mathcal{N}_1$ without mutation finds the global optimum of the Maxsat instance only with probability $2^{-\Theta(n)}$.*

PROOF. The lower bound on the success probability follows trivially from the fact that random initialization creates the global optimum with probability 2^{-n}.

For the upper bound, Chernoff bounds yield that the probability to start with x such that $2 \leq |x|_1 < 2n/3$ is $1-2^{-\Omega(n)}$. In that case flipping a single 1-bit increases the fitness as long as the offspring has at least two 1-bits left. Since this bit cannot flip back to 1, VDS returns a local optimum with a single 1-bit.

Having reached a local optimum, the least fitness decrease is obtained by flipping the unique 1-bit to 0. However, this implies that 1^n cannot be reached. As all other search points have worse fitness, VDS again returns a local optimum. □

In contrast to the previous problems, this is a first example where mutation is essential to find the global optimum.

THEOREM 9. *The expected number of generations of iterated VDS with mutation probability $p_m = 1/n$ on the Maxsat instance is $O(n)$.*

PROOF. The first VDS a local or global optimum. If a local optimum with a single 1-bit is reached, mutation creates 0^n with probability $1/n \cdot (1-1/n)^{n-1} \geq 1/(en)$ and the following VDS reaches 1^n with probability 1. The expected number of generations for this event is at most en. □

7. DISCUSSION AND CONCLUSIONS

We have considered single instances for three combinatorial problems and shown that memetic algorithms with VDS drastically outperforms many popular trajectory-based algorithms like the (1+1) EA, iterated local search, and simulated annealing. The list of combinatorial problems where VDS is effective is not complete. Similar analyses, using techniques from Section 3, can be performed e. g. for Graph Bisection, Maximum Clique, and Vertex Cover. We have chosen Mincut, Knapsack, and Maxsat since they resemble typical and well-known problems from different classes of problems: cutting, packing, and constraint optimization.

Furthermore, these three problems pose different challenges for randomized search heuristics. The Mincut instance yields a multimodal landscape with symmetric slopes. A search heuristic typically cannot tell in advance which hill might contain a global optimum. This secret is not revealed

until the algorithm climbs to the top of the hill and then it may have to climb down a long distance. For Maxsat the fitness landscape is deceptive, leading typical heuristics away from the global optimum. For the Knapsack instance we exploited that, from a macro-perspective, optimization is like searching for a needle in a haystack. All packings with the same number of objects have similar fitness, but only those without (or only few) big objects are promising. From a micro-perspective the instance is even worse since it gives deceptive hints towards big objects.

Iterated VDS is successful on these problem instances. This is partly due to the fact that VDS can cope with deceptive functions as it always encounters the bit-wise complement of the current search point. One may argue that iterated VDS is no more than a hill-climber tailored towards deceptive functions, like for example a hill-climber sampling around $\overline{x}$ in addition to the current population x. However, the Mincut instance cannot be optimized by such a specialized strategy. Another argument is that for Mincut and Maxsat VDS after some time discovers a positive gradient towards the global optimum and then is able to reach it "on its own", without the tabu mechanism. Finally, VDS is robust w. r. t. modifications of the instance. In the Knapsack instance the global and the local optimum are complementary. However, if we add some new objects with low profit and large weight a simple algorithm for deceptive functions fails. We conclude that VDS is more powerful than an algorithm tailored towards deceptive functions.

However, we cannot conclude that memetic algorithms with VDS are, in general, superior to common trajectory-based algorithms. The perspective taken in this work is one-sided as we only presented instances where memetic algorithms with VDS perform well, compared to common search strategies. It may be possible to find instances where memetic algorithms with VDS perform badly. Moreover, theory should not be restricted to single instances. We therefore regard the presented analyses as appetizers on the usefulness of memetic algorithms in combinatorial optimization from a theoretical perspective. We are still in need of a complete lunch, that is, broader results for important classes of instances for combinatorial problems to bring forward the theoretical understanding of hybrid algorithms.

Acknowledgment

The author thanks Carsten Witt for comments on a draft version.

8. REFERENCES

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[2] S. Droste, T. Jansen, and I. Wegener. A natural and simple function which is hard for all evolutionary algorithms. In *Proc. of IECON '2000*, pages 2704–2709. IEEE Press, 2000.

[3] S. Fischer. A polynomial upper bound for a mutation-based algorithm on the two-dimensional Ising model. In *Proc. of GECCO '04*, pages 1100–1112. Springer, 2004.

[4] T. Friedrich, P. S. Oliveto, D. Sudholt, and C. Witt. Theoretical analysis of diversity mechanisms for global exploration. In *Proc. of GECCO '08*, to appear.

[5] O. Giel and I. Wegener. Evolutionary algorithms and the maximum matching problem. In *Proc. of STACS '03*, pages 415–426. Springer, 2003.

[6] W. E. Hart, N. Krasnogor, and J. E. Smith, editors. *Recent Advances in Memetic Algorithms*. Springer, 2004.

[7] H. Ishibuchi, T. Yoshida, and T. Murata. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223, 2003.

[8] T. Jansen and I. Wegener. Real royal road functions: where crossover provably is essential. *Discrete Applied Mathematics*, 149(1-3):111–125, 2005.

[9] T. Jansen and I. Wegener. A comparison of simulated annealing with a simple evolutionary algorithm on pseudo-boolean functions of unitation. *Theor. Comput. Sci.*, 386(1-2):73–93, 2007.

[10] M. Jerrum and G. B. Sorkin. The metropolis algorithm for graph bisection. *Discrete Appl. Math.*, 82(1–3):155–175, 1998.

[11] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Tech J.*, 49(2):291–307, 1970.

[12] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[13] H. R. Lourenço, O. Martin, and T. Stützle. Iterated local search. In *Handbook of Metaheuristics*, pages 321–353. Kluwer Academic Publishers, Norwell, MA, 2002.

[14] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & OR*, 24(11):1097–1100, 1997.

[15] P. S. Oliveto, J. He, and X. Yao. Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. *International Journal of Automation and Computing*, 4(3):281–293, 2007.

[16] C. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[17] M. Stoer and F. Wagner. A simple min cut algorithm. In *Proc. of ESA '94*, pages 141–147, 1994.

[18] D. Sudholt. Crossover is provably essential for the Ising model on trees. In *Proc. of GECCO '05*, pages 1161–1167. ACM Press, 2005.

[19] D. Sudholt. Local search in evolutionary algorithms: the impact of the local search frequency. In *Proc. of ISAAC '06*, pages 359–368. Springer, 2006.

[20] D. Sudholt. On the analysis of the (1+1) memetic algorithm. In *Proc. of GECCO '06*, pages 493–500. ACM Press, 2006.

[21] I. Wegener. *Complexity Theory – Exploring the Limits of Efficient Algorithms*. Springer, 2005.

[22] I. Wegener. Simulated annealing beats metropolis in combinatorial optimization. In *Proc. of ICALP '05*, pages 589–601, 2005.

[23] C. Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *Proc. of STACS '05*, pages 44–56. Springer, 2005.