# Computing Minimum Cuts by Randomized Search Heuristics *


Frank Neumann
Algorithms and Complexity
Max-Planck-Institut für Informatik
Saarbrücken, Germany
fne@mpi-inf.mpg.de

Joachim Reichel
Institut für Mathematik
TU Berlin
Berlin, Germany
reichel@math.tu-berlin.de

Martin Skutella
Institut für Mathematik
TU Berlin
Berlin, Germany
skutella@math.tu-berlin.de



## ABSTRACT

We study the minimum $s$-$t$-cut problem in graphs with costs on the edges in the context of evolutionary algorithms. Minimum cut problems belong to the class of basic network optimization problems that occur as crucial subproblems in many real-world optimization problems and have a variety of applications in several different areas. We prove that there exist instances of the minimum $s$-$t$-cut problem that cannot be solved by standard single-objective evolutionary algorithms in reasonable time. On the other hand, we develop a bicriteria approach based on the famous MaxFlow-MinCut Theorem that enables evolutionary algorithms to find an optimum solution in expected polynomial time.



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

**General Terms:** Theory, Algorithms, Performance

**Keywords:** evolutionary algorithms, minimum $s$-$t$-cuts, multi-objective optimization, randomized search heuristics


## 1. INTRODUCTION

Metaheuristics such as evolutionary algorithms, ant colony optimization, and local search methods are known to be good problem solvers for a wide range of real-world optimization problems. Empirical tests confirm that they provide high-quality solutions within reasonable time for many such problems. Understanding the success of these metaheuristics from a theoretical point of view has gained increasing interest in recent years and is an ongoing challenge.

A lot of progress has been made in analyzing simple evolutionary algorithms with respect to their runtime behavior on artificial pseudo-boolean functions [4, 8] as well as some well-known combinatorial optimization problems [7, 13, 14, 15, 17, 18]. We contribute to this line of research and study the minimum $s$-$t$-cut problem in a given graph with weights on the edges. This is one of the basic, classical problems in combinatorial optimization, operations research, and computer science [2]. It is well known that the problem of computing a minimum $s$-$t$-cut can be solved in polynomial time and is closely related to the problem of computing a maximum flow in a given graph. Besides the classical $s$-$t$-cut problem, there are many other variants of cutting problems some of which are NP-hard. Examples are the maximum cut problem or the minimum multicut problem [9]. Evolutionary algorithms have produced good results for various kinds of difficult cutting problems [5, 12, 16].

We start by considering two single-objective models for the minimum $s$-$t$-cut problem in Section 2. One is node-based, the other one is edge-based. In the node based approach we are searching for a partitioning of the vertices into two subsets, one containing $s$ and the other containing $t$, such that the cost of the edges connecting the $s$- to the $t$-side of the cut is minimal. In the edge based approach we search for a subset of edges of minimal costs such that the deletion of those edges disconnects $t$ from $s$, i. e., the chosen edges constitute a cut.

It turns out that the two mentioned single-objective approaches do not lead to an efficient optimization process for basic evolutionary algorithms. We present classes of graphs with the following undesired property: Once the evolutionary algorithm has found some suboptimal $s$-$t$-cut, it is extremely unlikely that this solution will eventually be turned into an optimal solution within a polynomial number of iterations. The reason is that in the search space consisting of all $s$-$t$-cuts there is a second-best (suboptimal) solution such that the globally optimal solution is far away and thus very unlikely to be reached within a single step. Moreover, due to the special structure of the considered graphs, the evolutionary algorithm gets easily trapped in this second-best locally optimal solution.

Afterwards, in Section 3, we examine an edge-based multi-objective model of the problem that takes the cost of a subset of edges as well as the remaining $s$-$t$-flow value into account that can be sent after removing the chosen edges. This trick helps to somehow enlarge the actual search space by enhancing infeasible edge sets (whose removal does not disconnect $t$ from $s$). The enlarged search space no longer allows for the undesired situation in the single-objective approach discussed above.

In order to evaluate a subset of edges with respect to the maximum $s$-$t$-flow value after deletion of those edges, we assume that the evolutionary algorithm has access to an oracle that can compute the maximum flow value in a graph. Due to the close relation of maximum $s$-$t$-flows and minimum $s$-$t$-cuts, this assumption seems to be questionable at first sight. We therefore discuss this issue in some more detail in the following.

We first argue from a theoretical point of view. While an explicitly given maximum $s$-$t$-flow (specified by the flow


*This work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Center "Computational Intelligence" (SFB 531).

value on every edge of the graph) directly exposes a minimum $s$-$t$-cut, the maximum flow value alone does not contains any structural information about a minimum cut besides the minimum cut capacity. In particular, having access to such an oracle does not render the minimum cut problem entirely trivial.

From a more practical point of view, having access to such a maximum flow oracle seems reasonable in certain situations. Consider, for example, a network of water or oil pipelines. When a leak occurs at some point $t$ of the network, enough pipeline connections have to be cut off by using stop-cocks such that no more liquid leaks from the system. On the other hand, it is desirable to keep the number of inactivated pipeline connections at a minimum in order to keep the negative impact small. In the described scenario, after cutting off some edges, the remaining flow out of the leak can be easily observed and is actually the crucial basis for further decision-making.

Finally, in contrast to the basic minimum $s$-$t$-cut problem considered here, in more complex settings the complexity of a minimum cut computation and the related maximum flow computation can be considerably different. Consider for a example a multicommodity flow setting with $k$ source-sink pairs $(s_i, t_i)$, $i = 1, \ldots, k$. Here, a maximum multicommodity flow can be computed in polynomial time while the problem to find a set of edges of minimum cost that disconnects every sink $t_i$ from its associated source $s_i$, $i = 1, \ldots, k$, is NP-hard [3]. It is therefore reasonable to assume that maximum multicommodity flow computations are used as subroutines when trying to compute a minimum cut disconnecting all source-sink pairs. As a final example we mention length-bounded flows and cuts. Also in this situation, the maximum flow value is considerably easier to obtain than a minimum cut [1].

We continue with the discussion of the result presented in Section 3. As both criteria (the cost of chosen edges and the remaining flow value) admit a number of function values that is exponential in the input size, the Pareto front explored by the evolutionary algorithm is of exponential size and we investigate a multi-objective evolutionary algorithm that uses the concept of $\varepsilon$-dominance introduced in [10]. This concept leads to a partitioning of the two-dimensional objective space into a certain number of boxes. For each box at most one search point is archived. The size of the boxes is determined by a parameter $\varepsilon$ which has to be chosen according to the considered problem. We show that this algorithm performs well for a wide range of $\varepsilon$-values. In particular, we show that a minimum $s$-$t$-cut can be computed in expected polynomial time using the multi-objective approach.

The outline of the paper is as follows. In Section 2, we analyze the single-objective approaches to the minimum cut problem and show that they do not lead to an efficient optimization process. In Section 3, we present the multi-objective model and prove that the expected optimization time of this approach is polynomial. Finally, we finish with some conclusions.

# 2. SINGLE-OBJECTIVE APPROACH

We consider the following problem. Given a connected directed graph $G = (V, E)$ on $n + 2$ vertices and $m$ edges and a cost function $c : E \to \mathbb{N}_+$ that imposes positive integer weights on the edges. Two nodes $s, t \in V$ are distinguished. We call $s$ the source node and $t$ the target node.

A $s$-$t$-cut $S \subseteq E$ is a set of edges such that there is no path from $s$ to $t$ when the edges of $S$ are deleted from $E$. The cost of a subset of $E$ is defined as the sum of the costs of its elements. The goal is to find an $s$-$t$-cut $S \subseteq E$ of minimum cost. We denote by $c_{\max} = \max_{e \in E} c(e)$ the largest cost among all edges.

First, we examine two single-objective approaches. Here, we consider the well-known (1+1) EA working on bit strings of length $n$. New search points are obtained by flipping each bit of the current search point with probability $1/n$. The algorithm can be described as follows.

Algorithm 1. *(1+1) EA*

1. *Choose an initial search point $x \in \{0, 1\}^n$ uniformly at random.*
2. *Repeat*
   - *create an offspring $x'$ by flipping each bit of $x$ with probability $1/n$.*
   - *if $f(x') \leq f(x)$, $x := x'$.*

For our investigations, we are interested in the number of fitness evaluations to reach an optimal search point. This is called the optimization time of the considered algorithm. Often the expectation of this value is analyzed and called the expected optimization time.

## 2.1 Node-based Search

We first investigate the (1+1) EA that searches for a partitioning of the vertices such that the edges crossing the two partitions constitute a minimum cut.

The search space is $\{0, 1\}^n$, i. e., each bit of a search point $x$ corresponds to one vertex of $V \setminus \{s, t\}$. If $x_i = 0$, the vertex $v_i$ is on the same side of the cut as the source $s$ whereas $x_i = 1$ assigns $v_i$ to the target $t$. Let $S = \{s\} \cup \{v_i | x_i = 0\}$ and $T = \{t\} \cup \{v_i | x_i = 1\}$.

The fitness of a search point $x$ is given by

$$cost(x) = \sum_{e \in E \cap (S \times T)} c(e),$$

which computes the sum of the cost of all edges leading from $S$ to $T$. Note, that each search point constitutes a cut, i. e., there are no infeasible solutions using this approach.

In the following, we present a class of instances where the (1+1) EA in the described setting is not able to find a minimum cut in polynomial time with high probability. To simplify the presentation we use real-valued costs on the edges. However, an appropriate scaling can be used to come up with instances where the costs are positive integers and the following results also hold.

The example is based on graphs $G_k$ that are used as building blocks (see Figure 1). The graph $G_k$ consists of a path of $k$ nodes (excluding $s$ and $t$) connected by edge pairs. The costs on the edge pairs are increasing from 1 to $k$. In addition, the very last edge pair has cost 0, such that assigning all nodes to the source $s$ constitutes the unique minimum cut of cost 0. On the other hand, assigning all nodes to the target $t$ is a local optimum of cost 1.

For simplicity, we assume that the bits of a vector from $\{0, 1\}^k$ are in the same order as the corresponding vertices

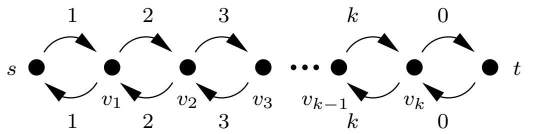


**Figure 1.** Graph $G_k$

on the path from $s$ to $t$. For example, $cost(0^{k-1}1) = k$. Furthermore, $cost(0^k) = 0$ and $cost(1^k) = 1$.

We define the notion of a block of bits as follows. A block is a set of consecutive bits that have the same value. The length of a block is the number of its bits. For example, $x = 0^k$ consists of one single block of 0's of length $k$. In the following, the right-most block of a bit string will play an important role, and we use $len(x)$ to define the length of that block.

First we prove two simple observations that will be needed later.

LEMMA 1. *Let $x$ be a search point with $len(x) \geq 2$. Provided that only 1-bit flips occur, all future accepted search points $x'$ satisfy $len(x') \geq len(x)$.*

PROOF. If the left-most bit of the right-most block is flipped, an edge of cost $k + 1 - len(x)$ is removed from the cut, whereas an edge of cost $k + 2 - len(x)$ is being added. Hence, $cost(x') = cost(x) + 1$ and $x'$ is not accepted. If the right-most bit of the right-most block is flipped, an edge of cost $k$ is added to the cut, whereas at most one edge of cost 0 is removed from the cut. Again, $cost(x') > cost(x)$ and $x'$ is not accepted. If one of the inner bits of the right-most block is flipped, two edges of cost at least $k+2-len(x)$ and $k+3-len(x)$ are added to the cut, and no edge is removed. This implies $cost(x') > cost(x)$ and $x'$ is not accepted. □

In other words: If $x = *00$ holds, this property is maintained for all future accepted search points (provided that only 1-bit flips occur). Similar for $x = *11$.

LEMMA 2. *Let $len(x) < k$. Flipping the bit left of the right-most block increases $len(x)$ by at least 1 and decreases $cost(x)$ by at least 1.*

PROOF. Let $i$ denote the index of the bit left to the right-most block. Let $x'$ denote the bit string obtained from $x$ by flipping $x_i$. Assume $x = *0$, this implies $x_i = 1$.

If $x_{i-1} = x'_{i-1} = 0$, flipping $x_i$ excludes the edges $(v_{i-1}, v_i)$ and $(v_{i+1}, v_i)$ from the cut, i. e., $cost(x') = cost(x) - i - (i + 1) \leq cost(x) - 1$. If $x_{i-1} = x'_{i-1} = 1$, flipping $x_i$ excludes the edge $(v_{i+1}, v_i)$ from the cut, whereas the edge $(v_i, v_{i-1})$ is included. Hence, we have $cost(x') = cost(x) - (i+1) + i = cost(x) - 1$. The case $x = *1$ is similar. □

Now we describe the construction of the graph $G_{k,\ell}$ based on $G_k$. Consider $\ell$ copies of $G_k$ and merge all copies of $s$. Similarly, merge all copies of $t$. The resulting graph has $k\ell + 2$ nodes and $2\ell(k + 1)$ edges. The $j$-th copy of $G_k$ will be denoted by $G^j$. The value $cost^j(x)$ corresponds to the total cost caused by the edges in $G^j$ leading from $S$ to $T$.

For the lower bound on the running time of the (1+1) EA we choose $k = \Theta(n^{1/10})$ and $\ell = \Theta(n^{1/10})$. Furthermore, we add $n - k\ell$ vertices adjacent to $t$. The resulting graph is called $G'_{k,\ell}$ (see Figure 2). In the following we distinguish between the original $G_{k,\ell}$ (called chain part) and the star part. All edges in the star part have cost $1/n$. Adding the star part has the consequence that steps flipping nodes in the flow part become more unlikely. As there are $\Theta(n^{1/5})$ nodes in the chain part but $\Theta(n)$ nodes in the star part, steps flipping exactly $i$ nodes in the chain part, $i$ a constant, have probability $\Theta(n^{-(4/5)i})$ (using similar counting arguments as in [15]).

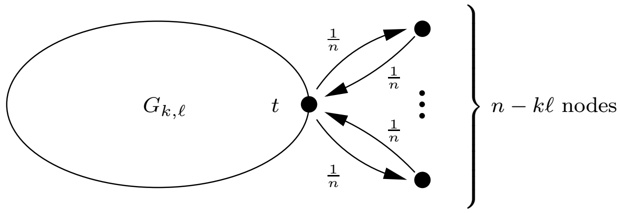


**Figure 2.** Graph $G'_{k,\ell}$

THEOREM 1. *With probability $1 - o(1)$, the optimization time of the (1+1) EA on $G'_{k,\ell}$ is $2^{\Omega(n^{1/10})}$.*

PROOF. We consider a typical run consisting of different phases of length $n^{7/5}$ and show that a local optimal solution which is not globally optimal is reached with probability $1 - o(1)$. As the chain part consists only of $\Theta(n^{1/5})$ nodes while the total number of nodes is $n$ (excluding $s$ and $t$), mutation steps flipping at least two bits in the chain part do not occur with probability $1 - O(n^{-8/5}n^{7/5}) = 1 - o(1)$ within these phases. Let $x^j$ be the part of a search point $x$ which consists of the bits corresponding to the nodes of $G^j$.

CLAIM 1. *With probability $1 - o(1)$, after $n^{7/5}$ steps a search point $x$ has been obtained for which the following two statements hold.*

1. *For each $G^j$ either $x^j = *00$ or $x^j = *11$ holds.*
2. *For at least one $G^j$, $x^j = *11$ holds.*

PROOF. The probability that a fixed $x^j$ of the initial search point does not match $*11$ is 3/4. These probabilities are independent for each component $G^j$. Hence, the probability that there is no $j$, $1 \leq j \leq \ell$ such that $x^j$ of the initial search point matches $*11$ is $(3/4)^\ell$. Thus, the second statement holds with probability $1 - o(1)$ for the initial search point. By Lemma 1, the statement holds with at least the same probability at the end of the phase.

Consider any component $G^j$. If $x^j = *00$ or $x^j = *11$ for the initial search point, by the same lemma, this property holds at the end of the phase. Suppose $x^j = *01$ or $x^j = *10$. The probability that the two right-most bits of $x^j$ are not flipped within a phase of $n^{7/5}$ steps is at most $(1 - 1/n)^{(2n^{7/5})} = O\left(e^{-2n^{2/5}}\right)$.

There are $\ell = \Theta(n^{1/10})$ components which implies that with probability $1 - O\left(n^{1/10}e^{-2n^{2/5}}\right) = 1 - o(1)$, $x^j = *00$ or $x^j = *11$ holds for each $j$ at the end of the phase. □

CLAIM 2. *With probability $1 - o(1)$, after additional $n^{7/5}$ steps a search point $x$ has been obtained for which the following two statements hold.*

1. *For each $G^j$ either $x^j = 0^k$ or $x^j = 1^k$ holds.*
2. *For at least one $G^j$, $x^j = 1^k$ holds.*

PROOF. With probability $1 - o(1)$ only such mutation steps occur that flip no bits or exactly one bit in the chain part. Mutation steps that flip no bits in the chain part are irrelevant for the claim and can be ignored.

Now consider the mutation steps where exactly one bit in the chain part is flipped. Due to the choice of $1/n$ for the cost of the star edges, the fitness change caused by the star part is at most $(n - k\ell)/n < 1$. The fitness changes by at least 1 when flipping exactly one bit in the chain part. Hence, changes in the star part do not affect the statements given in Lemma 1 and Lemma 2.

There is a sequence of at most $n^{1/5}$ 1-bit flips in the chain part that results in a search point $x$ fulfilling the first statement: For each component $G^j$ flip the bit left to the right-most block. By Lemma 2, such steps are accepted. By Lemma 1, the length of the right-most block does not decrease. The probability of a particular 1-bit flip in the chain part in the next mutation step is at least $1/(2en)$. Hence, the expected time until a search point fulfilling the first statement is reached is upper bounded by $O(n^{6/5})$. Using Markov's inequality, the probability of having reached such a search point within a phase of $n^{7/5}$ steps is $1 - o(1)$.

By Claim 1, there is at least one component $G^j$ with $x^j = *11$ at the end of first phase. By Lemma 1 and the first statement, $x^j = 1^k$ holds at the end of the second phase. □

CLAIM 3. *With probability $1 - o(1)$, after additional $n^{7/5}$ steps a search point $x$ has been obtained for which the following three statements hold.*

1. *For each $G^j$ either $x^j = 0^k$ or $x^j = 1^k$ holds.*
2. *For at least one $G^j$, $x^j = 1^k$ holds.*
3. *All bits corresponding to nodes in the star part are set to 1.*

PROOF. After having reached a search point where for each $G^j$ either $x^j = 0^k$ or $x^j = 1^k$ holds, bit flips affecting the chain part are only accepted if they flip at least $k$ flow nodes. This is exponentially unlikely during a phase of $n^{7/5}$ steps. Hence the first two statements are fulfilled at the end of the phase.

Within this phase all bits corresponding to star nodes are set to 1 with probability $1 - o(1)$ using similar fitness layer arguments as before. □

After having reached a search point where the three properties of the preceding claim hold, we consider one fixed component $G^j$ with $x^j = 1^k$. This component can only be turned into an optimal component by flipping all bits of $G^j$ in a single mutation step. The probability for this event is $O(n^{-k})$. The expected waiting time for such a step is $\Omega(n^k) = 2^{\Omega(k \log n)}$. Using Markov's inequality once more, the optimization time is $2^{\Omega(k)}$ with probability $1-o(1)$ as all failure probabilities during our typical run have been shown to be $o(1)$. □

An integral cost vector can be obtained by multiplying all edge costs by $n$. Theorem 1 also applies to the modified cost vector.

We want to remark that this result also applies to undirected graphs. A pair of oppositely directed edges of equal cost behaves exactly as a single, undirected edge of the same cost.

## 2.2 Edge-based Search

Now we consider an approach that searches for a set of edges which represents a minimum cut. Therefore we work with bit strings of length $m = |E|$ in the (1+1) EA. For a search point $x \in \{0,1\}^m$, the set $E(x) := \{e_i \in E \mid x_i = 1\}$ denotes the subset of $E$ corresponding to the 1's in $x$. Note, that not every search point represents an $s$-$t$-cut.

We consider the fitness function $f(x) := cost(x) + \alpha flow(x)$ for some $\alpha > 1$, where $cost(x) := \sum_{e \in E(x)} c(e)$ and $flow(x)$ denotes the maximum value of an $s$-$t$-flow in the graph $G(x) := (V, E \setminus E(x))$. The capacity of an edge $e \in E$ equals its cost $c(e)$. The fitness function is to be minimized. Note that $flow(x)$ vanishes if and only if $E(x)$ contains an $s$-$t$-cut of $G$. Hence, $flow(x)$ is a penalty term that penalizes bitstrings that do not correspond to a feasible solution. If $E(x)$ contains an $s$-$t$-cut of $G$, the fitness function equals the value of the corresponding cut. A factor $\alpha \leq 1$ is unsuitable, since the empty set would have smaller (or equal) fitness than the global optimum.

In the following, we present a class of instances for which (1+1) EA fails to explore a minimum cut in polynomial time with high probability. Again, we use real-valued costs on the edges to simplify the presentation. The instances are based on the graph $H_k$ (see Figure 3). $H_k$ consists of $k + 1$ edges from $s$ to $v$ with cost 1 and $k$ edges from $v$ to $t$ with cost $1 + \varepsilon$. Choosing $\varepsilon > \frac{1}{k}$ implies that the minimum $s$-$t$-cut of $H_k$ is given by the set of 1-edges. The set of $(1 + \varepsilon)$-edges is another cut of larger cost. Requiring $\varepsilon < \frac{2}{k}$ will turn out to be useful later.

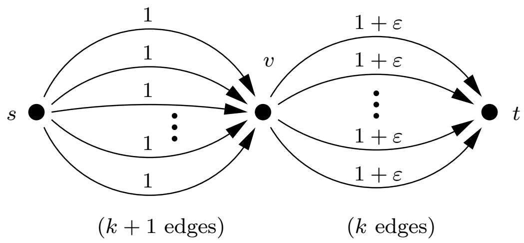


**Figure 3.** Graph $H_k$

Let $a(x)$ denote the cardinality of $E(x)$ intersected with the set of 1-edges, and let $b(x) := |E(x)| - a(x)$. Then $cost(x) = a(x) \cdot 1 + b(x) \cdot (1 + \varepsilon)$ and $flow(x) = \min\{k + 1 - a(x), (k - b(x))(1 + \varepsilon)\}$. Note that the global optimum corresponds to $a(x) = k + 1$ and $b(x) = 0$, whereas $a(x) = 0$ and $b(x) = k$ is a local optimum of strictly larger value.

First we show that $flow(x)$ does not depend on $a(x)$ provided that $b(x)$ is larger than $a(x)$.

PROPOSITION 1. *If $b(x) \geq a(x) + 1$, then $flow(x) = (k - b(x))(1 + \varepsilon)$. If $b(x) \leq a(x) - 1$, then $flow(x) = k + 1 - a(x)$.*

PROOF. If $b(x) \geq a(x) + 1$ holds, then $(k - b(x))(1 + \varepsilon) \leq (k - a(x) - 1)(1 + \varepsilon) \leq k - 1 - a(x) + \varepsilon k$. Since $\varepsilon < \frac{2}{k}$, we have $(k - b(x))(1 + \varepsilon) < k + 1 - a(x)$.

If $b(x) \leq a(x) - 1$ holds, then $(k - b(x))(1 + \varepsilon) \geq (k - a(x) + 1)(1 + \varepsilon) > k + 1 - a(x)$. □

In the case that only 1-bit flips occur the following slightly stricter precondition is maintained throughout the run of (1+1) EA.

Lemma 3. *If $b(x) \geq a(x)+2$ for some search point $x$, then this property also holds for all future accepted search points, provided that only 1-bit flips occur.*

Proof. Let $x'$ denote the search point constructed from $x$. Since $x'$ differs from $x$ by one bit, we have $b(x') \geq a(x')+1$. By Proposition 1, the $flow(\cdot)$ value depends solely on $b(\cdot)$, and it holds

$$flow(x') = flow(x) + \begin{cases} -(1+\varepsilon) & \text{if } b(x') = b(x)+1, \\ 0 & \text{if } b(x') = b(x), \\ (1+\varepsilon) & \text{if } b(x') = b(x)-1. \end{cases}$$

For the $cost(\cdot)$ component holds

$$cost(x') = cost(x) + \begin{cases} 1+\varepsilon & \text{if } b(x') = b(x)+1, \\ -(1+\varepsilon) & \text{if } b(x') = b(x)-1, \\ 1 & \text{if } a(x') = a(x)+1, \\ -1 & \text{if } a(x') = a(x)-1, \\ 0 & \text{otherwise}. \end{cases}$$

Note that $\alpha > 1$. Hence, we can summarize that $f(x') \leq f(x)$ holds if and only if $a(x') = a(x)-1$ or $b(x') = b(x)+1$. This implies that 1-bit-flips are only accepted if they decrease $a(x)$ or increase $b(x)$, and hence, $b(x') \geq a(x')+2$ holds if $x'$ is accepted. □

In a similar way we obtain the following result.

Lemma 4. *If $a(x) \geq b(x)+2$ for some search point $x$, then this property also holds for all future accepted search points, provided that only 1-bit flips occur.*

Proof. Let $x'$ denote the search point constructed from $x$. Since $x'$ differs from $x$ by one bit, we have $a(x') \geq b(x')+1$. By Proposition 1, the $flow(\cdot)$ value depends solely on $a(\cdot)$, and it holds

$$flow(x') = flow(x) + \begin{cases} -1 & \text{if } a(x') = a(x)+1, \\ 0 & \text{if } a(x') = a(x), \\ 1 & \text{if } a(x') = a(x)-1. \end{cases}$$

For the $cost(\cdot)$ component holds

$$cost(x') = cost(x) + \begin{cases} 1+\varepsilon & \text{if } b(x') = b(x)+1, \\ -(1+\varepsilon) & \text{if } b(x') = b(x)-1, \\ 1 & \text{if } a(x') = a(x)+1, \\ -1 & \text{if } a(x') = a(x)-1, \\ 0 & \text{otherwise}. \end{cases}$$

Note that $\alpha > 1$. Hence, we can summarize that $f(x') \leq f(x)$ holds if and only if $a(x') = a(x)+1$ or $b(x') = b(x)-1$. This implies that 1-bit flips are only accepted if they increase $a(x)$ or decrease $b(x)$, and hence, $a(x') \geq b(x')+2$ holds if $x'$ is accepted. □

Now we describe the construction of the graph $H_{k,l}$ based on $H_k$. Consider $l$ copies of $H_k$ and merge all copies of $s$. Similarly, merge all copies of $t$. The resulting graph has $l+2$ nodes and $l(2k+1)$ edges. The $j$-th copy of $H_k$ will be denoted by $H^j$. The values $a^j(x)$ and $b^j(x)$ correspond to the cardinality of $E(x)$ intersected with the set of 1- and $(1+\varepsilon)$-edges in $H^j$, respectively.

For the lower bound on the running time of the (1+1) EA we choose $k = \Theta(n^{4/10})$ and $\ell = \Theta(n^{1/10})$. Furthermore, we add a clique of $n-l-1$ vertices (one vertex being $t$). The resulting graph is called $H'_{k,\ell}$ (see Figure 4). In the following we distinguish between the original $H_{k,\ell}$ (called bundle part) and the clique part. All edges in the clique part have cost $\delta \leq (\alpha-1)/n^2$. Adding the clique has the consequence that steps flipping edges in the bundle part become more unlikely. As there are $\Theta(n^{1/2})$ edges in the bundle part but $\Theta(n^2)$ edges in the clique part, steps flipping exactly $i$ edges in the bundle part, $i$ a constant, have probability $\Theta(n^{-(3/2)i})$ (using similar counting arguments as in [15]).

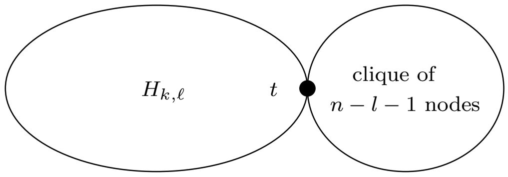


**Figure 4.** Graph $H'_{k,\ell}$

Theorem 2. *With probability $1-o(1)$, the optimization time of the (1+1) EA on $H'_{k,\ell}$ is $2^{\Omega(n^{1/10})}$.*

Proof. We consider a typical run consisting of different phases of length $n^{5/2}$ and show that a local optimal solution which is not globally optimal is reached with probability $1-o(1)$.

Claim 4. *With probability $1-o(1)$ for each $j$, $1 \leq j \leq \ell$, $|a^j(x) - b^j(x)| \geq 2$ holds for the initial search point $x$.*

Proof. Each component $H^j$ contains exactly $2k+1$ edges that can be either chosen or not. Let $x^j$ be the part of a search point $x$ which consists of the bits corresponding to these edges. Clearly, $x^j$ has $2k+1$ bits and the search space $X^j$ for $H^j$ is of size $2^{2k+1}$. In the following, we count the number of search points in $X^j$ where $|a^j(x) - b^j(x)| < 2$ holds. For $|a^j(x) - b^j(x)| = 0$ the number of search points in $X^j$ is given by

$$\sum_{i=0}^{k} \binom{k}{i}\binom{k+1}{i} \leq \binom{k}{k/2} \sum_{i=0}^{k} \binom{k+1}{i} = O(k^{-1/2}2^{2k+1})$$

For the case $a^j(x) - b^j(x) = -1$ we get

$$\sum_{i=1}^{k} \binom{k}{i}\binom{k+1}{i-1} = O(k^{-1/2}2^{2k+1})$$

For the case $a^j(x) - b^j(x) = 1$ we get

$$\sum_{i=0}^{k} \binom{k}{i}\binom{k+1}{i+1} = O(k^{-1/2}2^{2k+1})$$

Hence, the probability that $|a^j(x) - b^j(x)| < 2$ holds for the initial search point $x$ is upper bounded by

$$\frac{O(k^{-1/2}2^{2k+1})}{2^{2k+1}} = O(k^{-1/2}) = O(n^{-2/10})$$

There are $\ell = \Theta(n^{1/10})$ such components which implies that with probability $1 - O(n^{1/10} \cdot n^{-2/10}) = 1 - O(n^{-1/10}) = 1-o(1)$, $|a^j(x) - b^j(x)| \geq 2$ holds for each $j$, $1 \leq j \leq \ell$, of the initial search point $x$. □

Note that the property of Claim 4 does not only hold if the initial search point is chosen uniformly at random. For example, if the (1+1) EA is started from the empty set, the claimed property holds after an additional phase of $n^{5/2}$ steps with probability $1-o(1)$.

We consider phases consisting of $n^{5/2}$ steps. As the bundle part consists only of $\Theta(n^{1/2})$ edges while the total number of edges is $\Theta(n^2)$, mutation steps flipping at least two bits in the bundle part do not occur with probability $1-O(n^{5/2}n^{-3}) = 1-o(1)$ within these phases.

Claim 5. *With probability $1-o(1)$, after $n^{5/2}$ steps a search point $x$ has been obtained for which the following two statements hold.*

1. *For each $H^j$ either $b^j(x) = k$ and $a^j(x) = 0$ or $b^j(x) = 0$ and $a^j(x) = k+1$ holds.*
2. *For at least one $H^j$, $b^j(x) = k$ and $a^j(x) = 0$ holds.*

Proof. With probability $1-o(1)$ only such mutation steps occur that flip no bits or exactly one bit in the bundle part. Mutation steps that flip no bits in the bundle part are irrelevant for the claim and can be ignored.

Now consider the mutation steps where exactly one bit in the bundle part is flipped. Due to the choice of $\delta \leq \frac{\alpha-1}{n^2}$ for the cost of the clique edges, the fitness change caused by the clique part is at most $(n-l-1)(n-l-2)\delta < \alpha - 1$. The different cases examined in Lemma 3 and 4 show that the fitness changes by at least $\alpha - 1$ when flipping exactly one bit in the bundle part. Hence, changes in the clique part do not affect the statements given in Lemma 3 and 4.

Let

$$r^j = \begin{cases} k - b^j(x) + a^j(x) & \text{if } b^j(x) \geq a^j(x) + 2, \\ k + 1 - a^j(x) + b^j(x) & \text{if } a^j(x) \geq b^j(x) + 2. \end{cases}$$

and let $r = \sum_{j=1}^{\ell} r^j$ be the sum over the values $r^j$ for the components $H^j$. Due to Lemma 3 and 4, steps decreasing the value of $r$ are accepted while steps increasing the value of $r$ are rejected. The value $r$ decreases with probability at least $r/(em)$ in the next mutation step. Considering the different values of $r$ the expected time until a search point with $r = 0$ has been reached is upper bounded by

$$\sum_{r=1}^{n^{1/2}} (em/r) = O(m \log n).$$

Using Markov's inequality, the probability of having reached a search point where $r = 0$ holds within a phase of $n^{5/2}$ steps is $1-o(1)$.

After initialization $b^j(x) \geq a^j(x) + 2$ holds for at least one $H^j$ with probability $1-o(1)$. This implies that for this component the local optimum where $b^j(x) = k$ and $a^j(x) = 0$ is reached with probability $1-o(1)$ within the considered phase of $n^{5/2}$ steps. □

Claim 6. *With probability $1-o(1)$, after additional $n^{5/2}$ steps a search point $x$ has been obtained for which the following three statements hold.*

1. *For each $H^j$ either $b^j(x) = k$ and $a^j(x) = 0$ or $b^j(x) = 0$ and $a^j(x) = k+1$ holds.*
2. *For at least one $H^j$, $b^j(x) = k$ and $a^j(x) = 0$ holds.*
3. *All bits corresponding to edges in the clique part are set to 0.*

Proof. After having reached a search point where for each $H^j$ either $b^j(x) = k$ and $a^j(x) = 0$ or $b^j(x) = 0$ and $a^j(x) = k+1$ holds, bit flips affecting the bundle part are only accepted if they flip at least $2k+1$ bundle edges. This is exponentially unlikely during a phase of $n^{5/2}$ steps. Hence the first two statements are fulfilled at the end of the phase.

Within this phase all bits corresponding to edges in the clique part are set to 0 with probability $1-o(1)$ using similar fitness layer arguments as before. □

After having reached a search point where the three properties of the preceding claim hold, we consider one fixed component $H^j$ where $b^j(x) = k$ and $a^j(x) = 0$. This component can only be turned into an optimal component by flipping all bits of $H^j$ in a single mutation step. The probability for this event is $O(m^{-2k-1})$. The expected waiting time for such a step is $\Omega(m^{2k+1}) = 2^{\Omega((2k+1)\log m)}$. Using Markov's inequality once more, the optimization time is $2^{\Omega(k)}$ with probability $1-o(1)$ as all failure probabilities during our typical run have been shown to be $o(1)$. □

The example of this section can be modified as follows to obtain integral costs. Let $\frac{\beta}{\gamma}$ be a rational lower bound on $\alpha - 1$. We choose $\varepsilon := \frac{3}{2k}$ and $\delta := \frac{\beta}{\gamma}n^{-2}$. Finally we multiply all edge costs with $2k\gamma n^2$. The resulting cost vector is integral and the coefficients are polynomially bounded in the input size (for fixed $\alpha$). Theorem 2 also applies to the modified cost vector.

## 3. MULTI-OBJECTIVE APPROACH

In the multi-objective setting we consider an edge-based approach using the fitness function $f : \{0,1\}^m \mapsto \mathbb{N}_+^2, f(x) = (cost(x), flow(x))$, where $cost(x) := \sum_{e \in E(x)} c(e)$ and $flow(x)$ denotes the value of a maximum $s$-$t$-flow in $G(x) := (V, E \setminus E(x))$. Again, the capacity of an edge $e \in E$ equals its cost $c(e)$. Instead of combining $cost(x)$ and $flow(x)$ into one single value as in the single-objective setting, we consider both components separately. The objectives have to be minimized. Let $F := flow(0^m)$ denote the value of a maximum $s$-$t$-flow in $G$. Note that $F \leq C := m \cdot c_{\max}$. The goal is to find a search point $x$ with $f(x) = (F, 0)$.

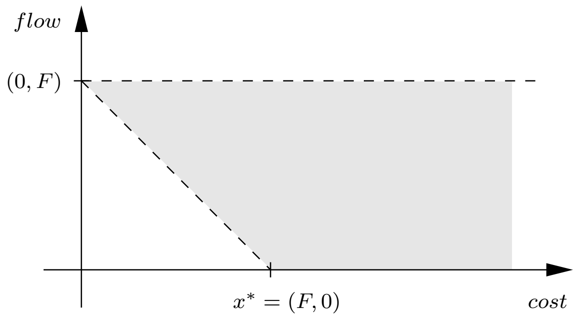


**Figure 5.** Objective space of the fitness function $f(x) = (cost(x), flow(x))$

The objective space is depicted in Figure 5. A simple observation about the structure of the search space is given in the following proposition.

PROPOSITION 2. *For any search point $x \in \{0,1\}^m$ it holds that $flow(x) + cost(x) \geq F$. Furthermore, $flow(x) + cost(x) = F$ if and only if $E(x)$ is a subset of some minimum cut.*

PROOF. Assume $flow(x) + cost(x) < F$ holds for some $x \in \{0,1\}^m$. By the definition of $flow(\cdot)$, there exists a maximum flow in $G(x)$ of value $flow(x)$. This maximum flow induces a minimum cut in $G(x)$ of the same value. The union of the edges crossing this cut and the edges in $E(x)$ is a cut for the original graph $G$ and the value of this cut is $flow(x) + cost(x) < F$. Since $F$ is the value of a maximum flow for $G$, this is a contradiction to the maximum-flow-minimum-cut theorem.

Now suppose $flow(x) + cost(x) = F$ holds. By the same arguments, there is a cut in $G$ that contains $E(x)$. Since the value of this cut equals $F$, it is a minimum cut. Conversely, suppose $E(x)$ is a subset of some minimum cut $S$. Then $S \setminus E(x)$ is a minimum cut in $G(x)$ with value $flow(x)$. Hence, $F = \sum_{e \in S} c(e) = \sum_{e \in E(x)} c(e) + \sum_{e \in S \setminus E(x)} c(e) = cost(x) + flow(x)$. □

We denote by $L = \{x \in \{0,1\}^m \mid flow(x) + cost(x) = F\}$ the set of search points whose objective vectors lie on the line given by the two objective values $(0, F)$ and $(F, 0)$. Due to Proposition 2 these search points represent subsets of edges of a minimum cut.

Examples for simple multi-objective evolutionary algorithms (MOEAs) that have been analyzed before are SEMO and GSEMO [6, 11, 14]. The GSEMO algorithm can be described as follows. Note that the fitness function $f$ is vector-valued and the $\leq$-comparison is to be understood component-wise.

ALGORITHM 2. *GSEMO (Global Simple Evolutionary Multi-objective Optimizer)*

1. *Choose $x \in \{0,1\}^m$ uniformly at random.*
2. *Determine $f(x)$ and initialize $P := \{x\}$.*
3. *Repeat*
   - *choose $x \in P$ uniformly at random.*
   - *create an offspring $x'$ by flipping each bit of $x$ with probability $1/m$.*
   - *let $P$ unchanged, if there is an $x'' \in P$ such that $f(x'') \leq f(x')$ and $f(x'') \neq f(x')$.*
   - *otherwise, exclude all $x''$ with $f(x') \leq f(x'')$ and add $x'$ to $P$.*

Note that the values of both components $cost(\cdot)$ and $flow(\cdot)$ of the fitness function can be exponential in the input size, which implies that GSEMO has to cope with a Pareto front of exponential size. As long as the costs on the edges are polynomially bounded in the number of vertices, we can show that GSEMO is able to compute a minimum cut in expected polynomial time when using the objective functions mentioned above.

THEOREM 3. *The expected time until GSEMO working on the fitness function $f$ constructs a minimum cut is $O(Fm(\log n + \log c_{\max}))$.*

PROOF. The population size is upper bounded by $F$ as GSEMO keeps at each time at most one solution per fixed flow value. First we consider the time until $0^m$ has been included into the population. Afterwards we study the time to reach a minimum cut afterwards.

We apply the method of the expected multiplicative cost decrease with respect to the cost value. Let $x \in P$ be the solution in the population with the smallest cost. Consider a mutation step that selects $x$ and performs an arbitrary 1-bit flip. Such a step is called a *good* step. The probability of a good step is lower bounded by $\Omega(1/F)$.

Each step removing an edge from the solution $x$ leads to a new solution with smaller cost and is accepted. Steps adding an edge to $x$ do not change the minimum cost. Therefore, a randomly chosen 1-bit flip decreases the minimum cost on average by a factor of at least $1 - 1/m$. This holds independently of previous steps. Hence, after $N$ good steps, the expected minimum cost value is bounded from above by $(1-1/m)^N \cdot cost(x)$. Since $cost(x) \leq C$, we obtain the upper bound $(1 - 1/m)^N \cdot C$.

If $N := \lceil (\ln 2) \cdot m \cdot (\log C) \rceil$, this bound is at most $\frac{1}{2}$. Since the minimum cost is not negative, by Markov's inequality, the probability that the bound is less than 1 is at least 1/2. The minimum cost is an integer implying that the probability of having found the search point $0^m$ is at least 1/2. Therefore, the expected number of good steps until the search point $0^m$ is found is bounded by $2N = O(m \log C) = O(m(\log n + \log c_{\max}))$. Since the probability of a good step is $\Omega(1/F)$, a total number of $O(Fm(\log n + \log c_{\max}))$ steps are needed to find the solution $0^m$.

Now we bound the time until a minimum cut has been constructed. Once again we apply the method of the expected multiplicative cost decrease, now with respect to the flow value. Let $x$ be the solution with the smallest flow value in $P \cap L$. Note, that solutions in $L$ are Pareto-optimal which implies that once a solution has been obtained for a specific objective vector, the population will always contain a solution for that objective vector until the end of the optimization process.

Consider a mutation step that selects $x$ and performs an arbitrary 1-bit flip. Such a step is called a *good* step. The probability of a good step is lower bounded by $\Omega(1/F)$. Due to Proposition 2, the solution $x$ is a subset of a minimum cut. A minimum cut and therefore a solution with objective vector $(F, 0)$ can be obtained by including the remaining edges of the corresponding minimum cut. Therefore, a randomly chosen 1-bit flip decreases the minimum flow value in $P \cap L$ on average by a factor of at least $1 - 1/m$.

Hence, after $N$ good steps, the expected minimum flow value is bounded from above by $(1-1/m)^N \cdot flow(x)$. Since $flow(x) \leq F \leq C$, we obtain the upper bound $(1 - 1/m)^N \cdot C$. Using the method of the multiplicative cost decrease the expected time until a minimum cut has been obtained is $O(Fm(\log n + \log c_{\max}))$ which proves the theorem. □

The upper bound given for GSEMO is pseudo-polynomial in the input size. In the following we consider a MOEA that ensures diversity by using the concept of $\varepsilon$-dominance introduced by LAUMANNS et al. [10]. Here the objective space is partitioned into axes-parallel boxes and each box includes at each generation at most one search point. It turns out that in our setting it is sufficient to partition the objective space with respect to the second objective. Hence, we use stripes instead of boxes.

We partition our objective space into stripes with respect to the flow value by using the function $b : \{0,1\}^m \mapsto N$ with

$b(x) := \left\lfloor \frac{\log(1+flow(x))}{\log(1+\varepsilon)} \right\rfloor$, where $\varepsilon > 0$ is a parameter that determines the size of the stripes. Let $B := \max_{x\in\{0,1\}^m} b(x)$.

Now the DEMO algorithm can be described as follows.

ALGORITHM 3. *DEMO (Diversity Evolutionary Multi-objective Optimizer)*

1. *Choose $x \in \{0,1\}^m$ uniformly at random.*
2. *Determine $f(x)$ and initialize $P := \{x\}$.*
3. *Repeat*
   - *choose $x \in P$ uniformly at random.*
   - *create an offspring $x'$ by flipping each bit of $x$ with probability $1/m$.*
   - *let $P$ unchanged, if there is an $x'' \in P$ such that $f(x'') \leq f(x')$ and $f(x'') \neq f(x')$ or if there is an $x'' \in P$ such that $b(x'') = b(x')$ and $cost(x'') + flow(x'') < cost(x') + flow(x')$.*
   - *otherwise, exclude all $x''$ where $f(x') \leq f(x'')$ or $b(x'') = b(x')$ and add $x'$ to $P$.*

Similar to the GSEMO algorithm, DEMO discards a new search point $x'$ if it is dominated by a search point $x'' \in P$ with different objective vector. Additionally, $x'$ is discarded if there is a search point $x'' \in P$ which falls in the same stripe and is closer to the line through $(0, F)$ and $(F, 0)$. If this is not the case, as before, all dominated search points in the population are removed. Additionally, we ensure that the population contains at most one search point for each stripe.

PROPOSITION 3. *The maximum population size of DEMO is bounded by $B = O(\varepsilon^{-1} \log C)$.*

PROOF. Since the $b(\cdot)$ value is a non-negative integer and the population contains at most one search point per stripe, the population size is bounded by $B$. We have $B \leq \log(1 + F)/\log(1+\varepsilon) \leq 2\log(1+F)/\varepsilon$. Since $F \leq C$, we obtain $b(x) = O(\varepsilon^{-1}\log C)$. □

LEMMA 5. *The expected time until DEMO working on the fitness function $f$ constructs a search point $x^* \in L$ is $O(m\varepsilon^{-1}(\log^2 n + \log^2 c_{\max}))$.*

PROOF. We proof the result by considering the expected multiplicative decrease of $flow(\cdot) + cost(\cdot)$.

Let $x = argmin_{z\in P} flow(z) + cost(z)$ be a solution whose sum of the flow and cost value is the smallest among all solutions in the current population $P$. Assume that $x \notin L$, i. e., $flow(x) + cost(x) = F$ does not hold.

Denote by $E'(x) \subseteq E(x)$ the set of edges chosen by $x$ that do not belong to a fixed minimum cut of $G$. Removing these edges leads to a solution $x^* \in L$ due to Proposition 2. Each of the 1-bit flips removing a single edge of $E'(x)$ leads to a solution $x'$ with $flow(x') + cost(x') \leq flow(x) + cost(x)$ as otherwise this would contradict Proposition 2.

We consider all possible 1-bit flips. The probability for carrying out an arbitrary 1-bit flip in the next step is $\Theta(1)$. All 1-bit flips regarding the edges of $E'(x)$ are accepted and in total lead to a solution $x^* \in L$. We do not consider the remaining 1-bit flips to measure the improvement towards a solution $x^* \in L$ and assume that they reduce the sum of the flow and cost value by zero.

Therefore, a randomly chosen 1-bit flip decreases the value $flow(x) + cost(x) - F$ on average by a factor of at least $1 - 1/m$. Using the method of the multiplicative cost decrease the expected number of steps until a search point $x^*$ with $flow(x^*) + cost(x^*) = F$ has been obtained is bounded by $2N = O(m \log C) = O(m(\log n + \log c_{\max}))$.

Since the population size is bounded by $B$, the probability of picking a search point $x$ that minimizes $flow(x)+cost(x)$ is $\Omega(1/B)$. Hence, the expected number of generations until a solution $x^* \in L$ has been obtained is upper bounded by

$$\begin{aligned} O(NB) &= O(mB\log C) = O(m\varepsilon^{-1}\log^2 C) \\ &= O(m\varepsilon^{-1}(\log^2 n + \log^2 c_{\max}))\,. \end{aligned}$$

This concludes the proof. □

In the following, we show that we can obtain from each search point $x \in L$ which does not describe a minimum cut a search point $x' \in L$ with $b(x') < b(x)$ by flipping a specific bit if the value of $\varepsilon$ is chosen in an appropriate way. Using this property we are able to show that DEMO is able to compute a minimum cut efficiently.

PROPOSITION 4. *Let $\varepsilon \leq 1/m$ and $x \in L$ be a search point with $flow(x) > 0$. Then there exists a 1-bit flip leading to a search point $x' \in L$ with $b(x') < b(x)$.*

PROOF. Let $y := flow(x)$. By Proposition 2, the set $E(x)$ is a subset of some minimum cut $E(x^*)$. Since $flow(x) > 0$, $E(x)$ is a proper subset. Hence, there exists a least one 1-bit flip leading to a search point $x'$ with $flow(x')+cost(x') = F$ and $flow(x') < flow(x)$. Among all such search points, consider a point $x'$ that minimizes $y' := flow(x')$.

Let $k := |E(x^*)| - |E(x)| \leq m$. Since $y'$ was minimal, $y' \leq (1 - \frac{1}{k})y$ holds. Since $\varepsilon \leq \frac{1}{m} \leq \frac{1}{k}$ and $k \leq y$, we have

$$\begin{aligned} (1+\varepsilon)(1+y') &\leq 1 + \varepsilon + (1+\varepsilon)\left(1 - \frac{1}{k}\right) y \\ &\leq 1 + \frac{y}{k^2} + \left(1 + \frac{1}{k}\right)\left(1 - \frac{1}{k}\right) y = 1 + y\,. \end{aligned}$$

This implies

$$1 + \frac{\log(1+y')}{\log(1+\varepsilon)} \leq \frac{\log(1+y)}{\log(1+\varepsilon)}\ ,$$

and finally $b(x') < b(x)$. □

THEOREM 4. *Choosing $\varepsilon \leq 1/m$, the expected time until DEMO working on the fitness function $f$ constructs a minimum cut is $O(m\varepsilon^{-2}(\log^2 n + \log^2 c_{\max}))$.*

PROOF. Due to Lemma 5 a search point $x \in L$ has been included into the population after an expected number of $O(m\varepsilon^{-1}(\log^2 n + \log^2 c_{\max}))$ steps. Hence, it is sufficient to consider the search process after having found a search point $x \in L$.

The archiving strategy of DEMO guarantees that each strip containing a search point from $P \cap L$ will contain such a (maybe different) search point in all future generations. Therefore, $\min_{x\in P\cap L} b(x)$ will never increase during the run of the algorithm.

Since the population size is bounded by $B$, the probability of picking a search point $x \in L$ with minimal $b$-value among the search points in $L$ is $\Omega(1/B)$. By Proposition 4, there exists at least one 1-bit flip leading to a search point $x' \in L$ with $b(x') < b(x)$. The probability to generate such a search point $x'$ is $\Omega(1/m)$. After at most $B$ such steps, the $b$-value is zero implying that we have found a minimum cut. Hence, the expected time to obtain a minimum cut is

$$O(B^2 m) = O(m\varepsilon^{-2}\log^2 C) = O(m\varepsilon^{-2}(\log^2 n + \log^2 c_{\max})).$$

This concludes the proof. □

Note, that the upper bound is $O(m^3(\log^2 n + \log^2 c_{\max}))$ for $\varepsilon = \Theta(1/m)$ and polynomial as long as $\varepsilon = 1/poly(m)$ for a polynomial $poly(m)$.

The analysis of Theorem 4 was split into two phases: the first phase ends with the construction of a solution $x \in L$. The total runtime is dominated by the runtime required for the second phase, which is a factor of $\varepsilon^{-1}$ larger than the runtime of the first phase. A better bound for the overall runtime can be proved using the following, slightly more powerful algorithm.

ALGORITHM 4. *Modified DEMO*

1. *Choose $x \in \{0,1\}^m$ uniformly at random.*
2. *Determine $f(x)$ and initialize $P := \{x\}$.*
3. *Repeat*
   - *choose $x \in P$ uniformly at random.*
   - *create an offspring $x'$ by flipping each bit of $x$ with probability $1/m$.*
   - *let $P$ unchanged, if there is an $x'' \in P$ such that $f(x'') \leq f(x')$ and $f(x'') \neq f(x')$ or if there is an an $x'' \in P$ such that $b(x'') = b(x')$ and $flow(x'') + cost(x'') < flow(x') + cost(x')$ or if there is an $x'' \in P$ such that $b(x'') = b(x')$ and $flow(x'')+cost(x'') = flow(x') + cost(x')$ and $flow(x'') < flow(x')$.*
   - *otherwise, exclude all $x''$ where $f(x') \leq f(x'')$ or $b(x'') = b(x')$ and add $x'$ to $P$.*

Compared to DEMO, a new rule for ties w.r.t. the sum of flow and cost value within the same stripe has been added. The modified version prefers solutions with smaller flow value, that is, solutions whose objective vector is close to the optimum $(F, 0)$. This ensures that $\min_{x \in P \cap L} flow(x)$ never increases during a run of the modified DEMO algorithm, which is not true for the original DEMO algorithm. This property is used to prove a tighter runtime bound.

THEOREM 5. *Choosing $\varepsilon \leq 1/m$, the expected time until modified DEMO working on the fitness function $f$ constructs a minimum cut is $O(m\varepsilon^{-1}(\log^2 n + \log^2 c_{\max}))$.*

PROOF. The modification does not affect the arguments in the proof Lemma 5. Hence, Lemma 5 also applies to the modified DEMO and it is sufficient to consider the search process after having found a search point $x \in L$.

Let $x$ be the solution with the smallest flow value in $P \cap L$. Consider a mutation step that selects $x$ and performs an arbitrary 1-bit flip. Such a step is called a *good* step. The probability of a good step is lower bounded by $\Omega(1/B)$. Due to Proposition 2, the solution $x$ is a subset of a minimum cut. A minimum cut and therefore a solution with objective vector $(F, 0)$ can be obtained by including the remaining edges of the corresponding minimum cut. Therefore, a randomly chosen 1-bit flip decreases the minimum flow value in $P \cap L$ on average by a factor of at least $1 - 1/m$.

Hence, after $N$ good steps, the expected minimum flow value is bounded from above by $(1-1/m)^N \cdot flow(x)$. Since $flow(x) \leq F \leq C$, we obtain the upper bound $(1-1/m)^N \cdot C$. Using the method of the multiplicative cost decrease the expected time until a minimum cut has been obtained is bounded by

$$O(Bm(\log n + \log c_{\max})) = O(m\varepsilon^{-1}(\log^2 n + \log^2 c_{\max}))$$

which proves the theorem. □

# 4. CONCLUSIONS

The computation of a minimum $s$-$t$-cut for a given weighted graph arises in several applications and many constrained variants are difficult to solve. We have studied how evolutionary algorithms can cope with this problem. Our investigations show that single-objective approaches fail to achieve optimal solutions. In contrast to this the proposed multi-objective approach points out the connection between the two contrasting objectives cost and feasibility. This approach leads to a polynomial runtime as long as the objective space is polynomially bounded. To overcome the latter problem we apply the concept of $\varepsilon$-dominance which lead to an expected polynomial runtime.

We believe that the gained insights might turn out to be useful for solving NP-hard variants of the problem in the context of multicommodity flow networks. In this context, computing a maximum flow is a relatively easy problem while finding a minimum cut is NP-hard. In a subsequent work we want to apply our bicriteria approach in order to solve the NP-hard minimum cut problem.

# References


[1] G. Baier. *Flows with Path Restrictions*. PhD thesis, TU Berlin, 2003.

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

[3] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23:864–894, 1994.

[4] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theor. Comput. Sci.*, 276:51–81, 2002.

[5] A. Duarte, Á. Sánchez, F. Fernández, and R. Cabido. A low-level hybridization between memetic algorithm and VNS for the max-cut problem. In *Proc. of GECCO'05*, pages 999–1006. ACM, 2005.

[6] O. Giel. Expected runtimes of a simple multi-objective evolutionary algorithm. In *Proc. of CEC 2003*, IEEE Press, pages 1918–1925, 2003.

[7] O. Giel and I. Wegener. Evolutionary algorithms and the maximum matching problem. In *Proc. of STACS 2003*, volume 2607 of *LNCS*, pages 415–426, 2003.

[8] T. Jansen and I. Wegener. Evolutionary algorithms - how to cope with plateaus of constant fitness and when to reject strings of the same fitness. *IEEE Trans. Evolutionary Computation*, 5(6):589–599, 2001.

[9] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, Berlin, 3rd edition, 2005.

[10] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10(3):263–282, 2002.

[11] M. Laumanns, L. Thiele, and E. Zitzler. Running time analysis of multiobjective evolutionary algorithms on pseudo-boolean functions. *IEEE Trans. Evolutionary Computation*, 8(2):170–182, 2004.

[12] K.-H. Liang, X. Yao, C. S. Newton, and D. Hoffman. A new evolutionary approach to cutting stock problems with and without contiguity. *Computers & OR*, 29(12):1641–1659, 2002.

[13] F. Neumann. Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *European Journal of Operational Research*, 181(3):1620–1629, 2007.

[14] F. Neumann and I. Wegener. Minimum spanning trees made easier via multi-objective optimization. *Natural Computing*, 5(3):305–319, 2006.

[15] F. Neumann and I. Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theor. Comput. Sci.*, 378(1):32–40, 2007.

[16] J. Puchinger, G. R. Raidl, and G. Koller. Solving a real-world glass cutting problem. In *Proc. of EvoCOP'04*, volume 3004 of *LNCS*, pages 165–176. Springer, 2004.

[17] J. Reichel and M. Skutella. Evolutionary algorithms and matroid optimization problems. In *Proc. of GECCO '07*, pages 947–954. ACM, 2007.

[18] C. Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *Proc. of STACS 2005*, volume 3404 of *LNCS*, pages 44–56, 2005.