Minimum Spanning Trees Made Easier Via Multi-Objective Optimization

Frank Neumann
Inst. für Informatik und Prakt. Mathematik
Christian-Albrechts-Univ. zu Kiel
24098 Kiel, Germany
fne@informatik.uni-kiel.de

Ingo Wegener *
FB Informatik, LS 2
Univ. Dortmund
44221 Dortmund, Germany
ingo.wegener@uni-dortmund.de

ABSTRACT

Many real-world problems are multi-objective optimization problems and evolutionary algorithms are quite successful on such problems. Since the task is to compute or approximate the Pareto front, multi-objective optimization problems are considered as more difficult than single-objective problems. One should not forget that the fitness vector with respect to more than one objective contains more information that in principle can direct the search of evolutionary algorithms. Therefore, it is possible that a single-objective problem can be solved more efficiently via a generalized multi-objective model of the problem. That this is indeed the case is proved by investigating the computation of minimum spanning trees.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Performance, Theory

Keywords

Multi-objective optimization, Working principles of evolutionary computing, Running time analysis

1. INTRODUCTION

Typical textbooks on optimization problems focus on single-objective optimization problems, see, e. g., Cormen, Leiserson, Rivest, and Stein (2001). The function f to be optimized is defined on a search space S and takes real values, i. e., $f\colon S \to \mathbb{R}$. For minimization problems on discrete search spaces S there may be many optimal search points $s \in S$ such that $f(s) \leq f(s')$ for all $s' \in S$ but only one optimal value $f_{\min} := \min\{f(s) \mid s \in S\}$. One is interested in the optimal value $f_{\min}$ and one optimal search point s.

In the case of multi-objective optimization problems the fitness function f is vector-valued, i. e., $f\colon S \to \mathbb{R}^k$. Since there is no canonical complete order on $\mathbb{R}^k$, one compares the quality of search points with respect to the canonical partial order on $\mathbb{R}^k$, namely $f(s) \leq f(s')$ iff $f_i(s) \leq f_i(s')$ for all $i \in \{1, \ldots, k\}$. A Pareto optimal search point s is a search point such that (in the case of minimization problems) $f(s)$ is minimal with respect to this partial order and all $f(s')$, $s' \in S$. Again there can be many Pareto optimal search points but they do not necessarily have the same fitness vector. The Pareto front consists of all fitness vectors $y = (y_1, \ldots, y_k)$ such that there exists a search point s where $f(s) = y$ and $f(s') \leq f(s)$ implies $f(s') = f(s)$. The problem is to compute the Pareto front and for each element y of the Pareto front one search point s such that $f(s) = y$. As in any case of optimization problems one may be satisfied with approximate solutions. This can be formalized as follows. For each element y of the Pareto front we have to compute a solution s such that $f(s)$ is close enough to y. Close enough is measured by an appropriate metric and an approximation parameter. In the single-objective case one switches to the approximation variant if exact optimization is too difficult. The same reason may hold in the multi-objective case. There may be another reason. The size of the Pareto front may be too large for exact optimization.

Sometimes, people try to turn multi-objective problems into single-objective ones, e. g., by optimizing a weighted sum of the fitness values of the single criteria. This may be useful in some applications but, in general, we do not obtain the information contained in the Pareto front and corresponding search points.

Multi-objective optimization has been an issue in operations research since a long time. Due to the typically high computational complexity of multi-objective problems the application of randomized search heuristics is a possibility to obtain satisfying solutions. Many variants of evolutionary algorithms specialized to multi-objective optimization problems have been developed and applied, for a survey see the monographs of Deb (2001) and Coello Coello, Van Veldhuizen, and Lamont (2002).

A conclusion from this discussion is that "multi-objective

*This work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Center "Computational Intelligence" (SFB 531) and by the German-Israeli Foundation (GIF) in the project "Robustness Aspects of Algorithms".

GECCO'05, June 25–29, 2005, Washington, DC, USA.

optimization is more (at least as) difficult than (as) single-objective optimization". This is true at least if the fitness values for the different criteria are "somehow independent". Without such an assumption there is no reason to believe in the conclusion above.

We discuss the following scenario. The considered problem is a single-objective problem. It is possible to add some further criteria such that the Pareto front of the newly created multi-objective optimization problem is not too large and such that the solution of the multi-objective problem includes the solution of the single-objective problem. Solving the multi-objective problem instead of the single-objective problem implies to compute the Pareto front instead of a single optimal value. Each considered search point contains more information than in the single-objective case since it contains also the fitness values for the additional criteria. At least in principle it is possible that this additional information improves the search behavior of evolutionary algorithms. This would imply that for solving difficult single-objective optimization problems one should also think about the possibility to model the problems as generalized multi-objective optimization problems.

The purpose of this paper is to prove that the considered scenario is not a fiction. We do not investigate artificial problems to support this claim but one of the combinatorial optimization problems contained in each textbook namely the computation of minimum spanning trees. (Nobody should expect that evolutionary algorithms computing minimum spanning trees beat the well-known problem-specific algorithms.)

In Section 2, we present the well-known evolutionary algorithms for multi-objective optimization that have been subject to a rigorous analysis of the expected optimization time. In Section 3, we introduce the two-objective variant of the minimum spanning tree problem which is subject of our investigations and distinguish it from other multi-objective variants of the minimum spanning tree problem. In Section 4, we prove upper bounds on the expected optimization time of some evolutionary algorithms for multi-objective optimization applied to our problems. It turns out that they are asymptotically smaller than lower bounds for the worst-case instances of simple evolutionary algorithms for the single-objective case. In order to investigate what happens for small problem dimensions and typical problem instances we have performed several experiments whose results are presented in Section 5. We finish with some conclusions.

2. EVOLUTIONARY ALGORITHMS FOR MULTI-OBJECTIVE OPTIMIZATION

The rigorous analysis of the expected optimization time of evolutionary algorithms is not easy. Most of such results are on simple evolutionary algorithms like the (1+1) EA (Droste, Jansen, and Wegener (2002)). This is even more true for multi-objective optimization. Therefore, we investigate and analyze the algorithm called SEMO (Simple Evolutionary Multi-Objective Optimizer) due to Laumanns et al.(2002). The algorithm starts with an initial solution $s \in \{0,1\}^n$. All non-dominated solutions are stored in the population P. In each step a search point from P is chosen uniformly at random and one bit is flipped to obtain a new search point s'. The new population contains for each non-dominated fitness vector $f(s)$, $s \in P \cup \{s'\}$, one corresponding search point and in the case that $f(s')$ is not dominated s' is chosen.

ALGORITHM 1. *SEMO*

1. *Choose an initial solution s.*
2. *Determine $f(s)$ and initialize $P := \{s\}$.*
3. *Repeat*
   - *choose $s \in P$ uniformly at random,*
   - *choose $i \in \{1,\ldots,n\}$ uniformly at random,*
   - *define $s' = (s'_1,\ldots,s'_n)$ by $s'_j = s_j$, if $j \neq i$, and $s'_i = 1 - s_i$,*
   - *determine $f(s')$,*
   - *let P unchanged, if there is an $s'' \in P$ such that $f(s'') \leq f(s')$ and $f(s'') \neq f(s')$*
   - *otherwise, exclude all s'' where $f(s') \leq f(s'')$ from P and add s' to P.*

In applications, we need a stopping criterion. Here we are interested in the expected number of rounds until $f(P) := \{f(s) | s \in P\}$ equals the Pareto front. This is called the expected optimization time. Note that the described algorithm differs from the original version of SEMO by replacing an individual s'' of P by s' if $f(s'') = f(s')$ holds. Applying our version of SEMO to a single-objective optimization problem, we obtain the algorithm known as RLS (randomized local search). All our results also hold for the original version of SEMO but it seems to be more typical for search heuristics to replace search points by other ones with the same quality (e.g., simulated annealing works this way). If SEMO starts with a search point s which is a local optimum, then $P = \{s\}$ forever. The use of this local mutation operator was motivated by the fact that this choice simplifies the analysis. Giel (2003) has generalized the investigations of Laumanns et al. (2002) and also Zitzler et al. (2003) by considering the usual mutation operator of evolutionary algorithms.

ALGORITHM 2. *GSEMO (Global SEMO)*
GSEMO works like SEMO but s' is defined in a different way. For each i, $s'_i = 1 - s_i$ with probability $1/n$ and $s'_i = s_i$ otherwise.

Note that GSEMO applied to single-objective optimization problems equals the well-known (1+1) EA. Hence, we compare SEMO and GSEMO with RLS and (1+1) EA.

3. A TWO-OBJECTIVE MODEL OF THE MINIMUM SPANNING TREE PROBLEM

An instance of the minimum spanning tree problem consists of an undirected graph $G = (V, E)$ with n vertices and m edges and a positive integer weight $w(e)$ for each edge. The problem is to find an edge set E' connecting all vertices of V with minimal total weight.

Neumann and Wegener (2004) have analyzed RLS (with 1-bit flips and 2-bit flips) and the (1+1) EA for the minimum spanning tree problem. They have used the following model of the problem. The search space is $S = \{0,1\}^m$ and $s \in S$ describes the edge set of all edges e_i where $s_i = 1$. Raidl

and Julstrom (2003) have shown that edge sets are appropriate for the minimum spanning tree problem. Neumann and Wegener (2004) have penalized edge sets which do not describe connected graphs (and in one model additionally edge sets containing cycles). They were able to prove the following results:

- The expected optimization time of RLS and the (1+1) EA is bounded by $O(m^2(\log n + \log w_{max}))$ where w_{max} is the largest weight of the considered graph.
- There are graphs with $m = \Theta(n^2)$ and $w_{max} = \Theta(n^2)$ such that the expected optimization time of RLS and the (1+1) EA equals $\Theta(m^2 \log n)$.

This is one of the first rigorous analyses of the expected optimization time of evolutionary algorithms on combinatorial optimization problems contained in textbooks. Previous results considered the computation of shortest paths (Scharnow, Tinnefeld, and Wegener (2002)) and maximum matchings (Giel and Wegener (2003)).

We discuss the reason for the expected optimization time of RLS and the (1+1) EA. If a search point describes a non-minimum spanning tree, one-bit flips are not accepted. Either the new search point describes an unconnected graph or a connected graph with a larger weight. We have to wait until a mutation step includes an edge and excludes a heavier one from the newly created cycle. The expected waiting time for a specified 2-bit flip equals $\Theta(m^2)$.

As already mentioned, the considered algorithms penalize the number of connected components. This motivates the following two-objective optimization model of the minimum spanning tree problem.

- The search space S equals $\{0,1\}^m$ for graphs on m edges and the search point s describes an edge set.
- The fitness function $f : S \to \mathbb{R}^2$ is defined by $f(s) = (c(s), w(s))$ where $c(s)$ is the number of connected components of the graph described by s and $w(s)$ is the total weight of all chosen edges.
- Both objectives have to be minimized.

We discuss some simple properties of this problem.

- The parameter $c(s)$ is an integer from $\{1, \ldots, n\}$.
- The first property implies that the populations of SEMO and GSEMO contain at most n search points and the Pareto front contains exactly n elements.
- The parameter $w(s)$ is an integer.

We have to be careful when discussing this problem. There exists another type of multi-objective minimum spanning tree problem. Each edge has k different types of weights, i. e., $w(e) = (w_1(e), \ldots, w_k(e))$. Unconnected graphs are penalized and the aim is to minimize $f(s)$ where s is not legal if s does not describe a connected graph and $f(s)$ is the sum of all $w(e_i)$ where $s_i = 1$, otherwise. Similarly to other optimization problems this multi-objective variant of a polynomially solvable problem is NP-hard (Ehrgott (2000)). This problem has been attacked in different ways, e. g., by Hamacher and Ruhe (1994). Zhou and Gen (1999) present experimental results for evolutionary algorithms and Neumann (2004) has analyzed which parts of the Pareto front can be obtained in expected pseudopolynomial time.

4. THE ANALYSIS OF THE EXPECTED OPTIMIZATION TIME

Our results hold for SEMO as well as for GSEMO. The essential steps are 1-bit flips. In the definition of SEMO and GSEMO we have not specified how to choose the first search point. We discuss two possibilities.

- The first search point is chosen uniformly at random. This is the typical choice for evolutionary algorithms.
- The first search point is $s = 0^m$ describing the empty edge set. This is quite typical, e. g., for simulated annealing.

Our analysis is simplified by knowing that P contains 0^m. Note that $f(0^m) = (n, 0)$ belongs to the Pareto front and 0^m is the only search point s with $c(s) = n$. First, we investigate the expected time until the population contains the empty edge set.

THEOREM 1. *Starting with an arbitrary search point the expected time until the population of SEMO or GSEMO contains the empty edge set is bounded above by* $O(mn(\log n + \log w_{max}))$.

PROOF. One might expect that we only have to wait until all edges of the initial search point s have been excluded. This is not true. It is possible that we accept the inclusion of edges since this decreases the number of connected components (although it increases the total weight). Later, we may exclude edges of the new search point s' without increasing the number of connected components. It is possible to construct a search point s'' which dominates s. Then s is eliminated and all search points in the population (perhaps only one) have more edges than s.

Hence, the situation is more complicated. Instead of the minimal number of edges of all search points in P we analyze the minimal weight of all search points in P. One search point s^* with minimal weight has the largest number of connected components (otherwise, the search point s^{**} with $c(s^{**}) > c(s^*)$ is dominated by s^* and will be excluded from P). We analyze $w(s^*)$. We have reached the aim of our investigations if $w(s^*) = 0$, since this implies $s^* = 0^m$. After initialization, $w(s^*) \leq W := w_1 + \cdots + w_m \leq m \cdot w_{max}$.

We only investigate steps where s^* is chosen for mutation. The probability of such a step is always at least $1/n$, since $|P| \leq n$. Hence, the expected time is only by a factor of at most n larger than the expected number of steps where s^* is chosen.

By renumbering, we may assume that s^* has chosen the first k edges. We investigate only steps flipping exactly one bit. This has probability 1 for SEMO and probability at least e^{-1} for GSEMO, where $e = 2.71\ldots$. These steps are accepted if they flip one of the first k edges. If the edge i is flipped, we obtain a search point whose weight is $w(s^*) - w_i$ and the minimal weight has been decreased by a factor of $1 - \frac{w_i}{w(s^*)}$. The average factor of the weight decrease equals

$$\frac{1}{m}\left(\sum_{1 \leq i \leq k}(1 - \frac{w_i}{w(s^*)}) + \sum_{k+1 \leq i \leq m} 1\right) = 1 - \frac{1}{m}$$

if the choice of a non-existing edge is considered as a weight decrease by a factor of 1. The result $1 - \frac{1}{m}$ does not depend on the population. After $M := \lceil (\ln 2) \cdot m \cdot (\log W + 1) \rceil$ steps

choosing the current s^*, the expected weight of the new s^* is bounded above by $(1-1/m)^M \cdot W \leq \frac{1}{2}$. Applying Markoff's inequality, the probability that $w(s^*) < 1$ is bounded above by 1/3 (or $1/2-\varepsilon$ for each $\varepsilon > 0$). Since weights are integers, $w(s^*) < 1$ implies $w(s^*) = 0$. The expected number of phases of length M until $w(s^*) = 0$ is less than 3. Hence, altogether the expected waiting time for $s^* = 0^m$ is bounded above by $3 \cdot n \cdot M = O(mn(\log n + \log w_{max}))$ for SEMO. The corresponding value for GSEMO is only by a factor of less than 3 larger. □

One may expect that this upper bound is not exact for many graphs and starting points.

THEOREM 2. *Starting with a population containing the empty edge set the expected optimization time of SEMO or GSEMO is bounded by $O(mn^2)$.*

PROOF. As long as the algorithm has not reached its goal we consider the smallest i such that the population contains for each j, $i \leq j \leq n$, a Pareto optimal search point s_j with $f(s_j) = (j, \cdot)$. This implies that the graph described by s_j consists of j connected components and has the minimal possible weight among all possible search points describing graphs with j connected components. After initialization, the population includes 0^m which has the smallest weight among all search points representing graphs with n connected components. Hence, i is well defined. The search point s_j is only excluded from the population if a search point s'_j with $f(s'_j) = f(s_j)$ is included in the population. Hence, the crucial parameter i can only decrease and the search is successful if $i = 1$.

Finally, we investigate the probability of decreasing i. It is well-known that a solution with $i-1$ components and minimal weight can be constructed from a solution with i components and minimal weight by introducing the lightest edge that does not create a cycle. Therefore, it is sufficient to choose s_i for mutation (probability at least $1/n$) and to flip exactly one bit concerning a lightest edge connecting two components in the graph described by s_i (probability at least $1/m$ for SEMO and at least $1/(em)$ for GSEMO). Hence, the expected waiting time to decrease the parameter i is bounded above by $O(nm)$. After at most $n-1$ of such events the search is successful. □

COROLLARY 1. *If the weights are bounded above by 2^n, SEMO and GSEMO find the Pareto front in the two-objective variant of the minimum spanning tree problem in an expected number of $O(mn^2)$ rounds independently from the choice of the first search point.*

For dense graphs, this bound beats the bound $O(m^2 \cdot \log n)$ for the application of RLS and the (1+1) EA to the single-objective variant of the minimum spanning tree problem.

5. EXPERIMENTAL RESULTS

The theoretical results are asymptotic ones. They reveal differences for worst-case instances and large m. We add experimental results that show what happens for typical instances and reasonable m. In order to compare randomized algorithms on perhaps randomly chosen instances one may compare the average run times, but these values can be highly influenced by outliers. We have no hypothesis about the probability distribution describing the random run time for constant input length. Hence, only parameter-free statistical tests can be applied. We apply the Mann-Whitney test (MWT) that ranks all observed run times. Small ranks correspond to small run times. If the average rank of the results of algorithm A_1 are smaller than those for A_2, MWT decides how likely it can be that such a difference or a larger one can occur under the assumption that A_1 is not more efficient than A_2. If the corresponding p-value is at most 0.05, we call the result significant, for 0.01 very significant, and for 0.001 highly significant. The statistical evaluation has been performed with the software SPSS (Version 11.5, see www.spss.com). The tables contain the considered class of graphs, the average rank AR of different algorithms and the p-value for the hypothesis that the algorithm with the smaller AR-value is likely to be faster.

The experiments consider the following graph classes.

- uniform$_n$: these are complete graphs with $m = \binom{n}{2}$ edges and the weights are chosen independently and uniformly at random from $\{1, \ldots, n\}$.
- uniformbd$_n$: each possible edge is chosen with probability $3/n$ leading to a small average degree of 3, unconnected graphs are rejected and the construction is repeated, the weights of existing edges are chosen as for uniform$_n$.
- plane$_n$: the n vertices are placed randomly on the points of the two-dimensional grid $\{1, \ldots, n\} \times \{1, \ldots, n\}$, the weight of an edge is the rounded Euclidean distance between the vertices.
- planebd$_n$: the n vertices are placed as for plane$_n$ but each edge is only considered with probability $3/n$ as for uniformbd$_n$.

These graph classes reflect different choices of weights (one non-metric and one metric one) and the possibility of dense and sparse graphs. Our algorithms are RLS, (1+1) EA, SEMO, and GSEMO. The index z denotes the case that the initial search point is the empty edge set (or all-zero string). Without an index the initial search point is chosen uniformly at random. The run time of RLS and the (1+1) EA denotes the number of fitness evaluations until a minimum spanning tree is constructed. The run time of SEMO and GSEMO denotes the number of rounds until, in one experiment, P contains a minimum spanning tree or until $f(P)$ equals the Pareto front. In each experiment the compared algorithms are considered for 100 runs leading to an average rank of 100.5.

We have analyzed the influence of the initial search point. First, we have considered the time until the Pareto front is computed. The results are shown in Table 1.

RESULT 1. *In 23 out of 24 experiments the variant starting with the empty edge set has the smaller AR-value. Only 8 results are significant, among them 5 very significant and 2 of these highly significant.*

If we are only interested in the computation of a minimum spanning tree, one may expect that one sometimes computes a minimum spanning tree without computing the empty edge set. Indeed, the influence of the choice of the initial search point gets smaller. For the classes uniform$_n$, $n = 4i$ and $3 \leq i \leq 11$, there is no real difference between

Table 1: Comparison of SEMO and GSEMO with different initial solutions until they have computed the Pareto front

Class	AR SEMO_z	AR SEMO	p-value	AR GSEMO_z	AR GSEMO	p-value
$uniform_{12}$	92.76	108.25	0.058	89.35	111.66	**0.006**
$uniform_{16}$	83.51	117.49	$<$ **0.001**	91.28	109.72	**0.024**
$uniform_{20}$	99.12	101.89	0.735	94.21	106.80	0.124
$uniform_{24}$	98.01	102.99	0.543	93.65	107.35	0.094
$uniform_{28}$	94.62	106.38	0.151	94.48	106.52	0.141
$uniform_{32}$	91.24	109.76	**0.024**	96.76	104.24	0.361
$plane_{12}$	81.61	119.39	$<$ **0.001**	88.14	112.86	**0.003**
$plane_{16}$	94.51	106.49	0.143	89.38	111.63	**0.007**
$plane_{20}$	97.17	103.83	0.416	95.15	105.85	0.191
$plane_{24}$	93.33	107.67	0.080	103.11	97.89	0.524
$plane_{28}$	90.58	110.43	**0.015**	93.09	107.91	0.070
$plane_{32}$	94.55	106.45	0.146	97.44	103.56	0.455

Table 2: Comparison of SEMO and GSEMO with their single-criteria counterparts on complete uniform and complete geometric instances

Class	AR RLS	AR SEMO	p-value	AR (1+1) EA	AR GSEMO	p-value
$uniform_{12}$	146.36	54.64	$<$ **0.001**	147.79	53.32	$<$ **0.001**
$uniform_{16}$	148.45	52.55	$<$ **0.001**	149.28	51.72	$<$ **0.001**
$uniform_{20}$	149.74	51.26	$<$ **0.001**	149.40	51.60	$<$ **0.001**
$uniform_{24}$	150.00	51.00	$<$ **0.001**	150.29	50.71	$<$ **0.001**
$uniform_{28}$	150.40	50.60	$<$ **0.001**	150.23	50.77	$<$ **0.001**
$uniform_{32}$	150.50	50.50	$<$ **0.001**	150.50	50.50	$<$ **0.001**
$plane_{12}$	141.43	59.58	$<$ **0.001**	145.04	55.96	$<$ **0.001**
$plane_{16}$	144.25	56.75	$<$ **0.001**	148.28	52.72	$<$ **0.001**
$plane_{20}$	149.47	51.53	$<$ **0.001**	149.54	51.46	$<$ **0.001**
$plane_{24}$	149.95	51.05	$<$ **0.001**	149.89	51.11	$<$ **0.001**
$plane_{28}$	150.40	50.60	$<$ **0.001**	150.36	50.64	$<$ **0.001**
$plane_{32}$	150.34	50.66	$<$ **0.001**	150.28	50.72	$<$ **0.001**

SEMO_z and SEMO, while the AR-values of GSEMO are in 8 of the 9 experiments smaller than for GSEMO_z. For the classes plane_n, $n = 4i$ and $3 \leq i \leq 11$, SEMO_z beats SEMO (7 cases) and GSEMO_z beats GSEMO (7 cases). We do not show the results in detail since they are not significant (with the exception of 3 out of 36 cases). The remaining experiments consider the more general case of an initial search point chosen uniformly at random.

We have not considered the worst-case instances for RLS and (1+1) EA presented by Neumann and Wegener (2004). This would be unfair against these algorithms. Nevertheless, the experiments of Briest et al. (2004) have indicated that, for n and m of reasonable size, dense random graphs are even harder than the asymptotic worst-case examples. This leads to the conjecture that SEMO beats RLS and GSEMO beats its counterpart (1+1) EA. Here, the run time measures the rounds until a minimum spanning tree is constructed. Table 2 proves that our conjecture holds for the considered cases. Note that the average rank of 100 runs of one algorithm is at least 50.5. In several experiments the AR-value of SEMO or GSEMO comes close to this value. For $n \geq 20$ all values are at most 51.6 and for small values of n the AR-values are smaller than 60.

RESULT 2. *It is highly significant for all considered graph classes and graph sizes that SEMO outperforms RLS and GSEMO outperforms the (1+1) EA.*

The theoretical analysis of the algorithms gives values of $O(m^2 \log n)$ for RLS and the (1+1) EA and $O(mn^2)$ for SEMO and GSEMO (if the weights are reasonably bounded). For complete graphs, $m = \Theta(n^2)$ and we get values $n^4 \log n$ vs. n^4. For sparse graphs, $m = \Theta(n)$ and we get values $n^2 \log n$ vs. n^3. Although these are only upper bounds, one may expect different results for the sparse graphs from uniformbd_n and planebd_n. Table 3 shows that this is indeed the case.

RESULT 3. *It is highly significant for* uniformbd_n *and* $n \geq 24$ *and for* planebd_n *and* $n \geq 16$ *(and the considered values of* n*) that RLS outperforms SEMO. Similar results hold for the (1+1) EA and GSEMO, but the results are highly significant only for large values of* n*, namely* $n \geq 32$ *for both graph classes.*

Note that the last group of experiments considers values of n up to 100.

Table 3: Comparison of SEMO and GSEMO with their single-criteria counterparts on uniform and geometric instances with bounded average degree

Class	AR RLS	AR SEMO	p-value	AR (1+1) EA	AR GSEMO	p-value
$uniformbd_{12}$	91.91	109.09	**0.036**	101.44	99.57	0.819
$uniformbd_{16}$	90.62	110.39	**0.016**	103.54	97.46	0.458
$uniformbd_{20}$	89.79	111.22	**0.009**	98.98	102.02	0.710
$uniformbd_{24}$	73.19	127.82	$<$ **0.001**	91.53	109.47	**0.028**
$uniformbd_{28}$	78.01	122.99	$<$ **0.001**	93.03	107.98	0.068
$uniformbd_{32}$	77.92	123.08	$<$ **0.001**	80.85	120.15	$<$ **0.001**
$uniformbd_{40}$	73.02	127.98	$<$ **0.001**	84.37	116.63	$<$ **0.001**
$uniformbd_{60}$	65.40	135.60	$<$ **0.001**	71.22	129.78	$<$ **0.001**
$uniformbd_{80}$	56.70	144.30	$<$ **0.001**	58.72	142.28	$<$ **0.001**
$uniformbd_{100}$	54.99	146.01	$<$ **0.001**	58.47	142.53	$<$ **0.001**
$planebd_{12}$	97.56	103.45	0.472	105.24	95.77	0.247
$planebd_{16}$	81.88	119.13	$<$ **0.001**	96.79	104.22	0.364
$planebd_{20}$	81.06	119.95	$<$ **0.001**	101.70	99.30	0.769
$planebd_{24}$	84.45	116.55	$<$ **0.001**	86.52	114.48	**0.001**
$planebd_{28}$	81.94	119.06	$<$ **0.001**	88.45	112.55	**0.003**
$planebd_{32}$	71.53	129.47	$<$ **0.001**	80.86	120.14	$<$ **0.001**
$planebd_{40}$	67.18	133.82	$<$ **0.001**	74.57	126.44	$<$ **0.001**
$planebd_{60}$	56.59	144.41	$<$ **0.001**	60.69	140.31	$<$ **0.001**
$planebd_{80}$	52.98	148.02	$<$ **0.001**	59.60	141.40	$<$ **0.001**
$planebd_{100}$	52.21	148.79	$<$ **0.001**	52.30	148.70	$<$ **0.001**

6. CONCLUSIONS

It has been investigated whether the multi-objective variant of a single-variant optimization problem can lead to more efficient optimization processes. This is indeed the case for the well-known minimum spanning tree problem and randomly chosen dense graphs. For sparse connected graphs it is better to use the single-objective variant of the problem. The results are obtained by a rigorous asymptotic analysis of the expected optimization time and by experiments on graphs of reasonable size.

7. ACKNOWLEDGEMENT

The authors thank Dirk Sudholt who performed the statistical tests with the SPSS software.

8. REFERENCES

[1] Briest, P., Brockhoff, D., Degener, B., Englert, M., Gunia, C., Heering, O., Jansen, T., Leifhelm, M., Plociennik, K., Röglin, H., Schweer, A., Sudholt, D., Tannenbaum, S., and Wegener, I. (2004). Experimental supplements to the theoretical analysis of EAs on problems from combinatorial optimization. In Proc. of the 8th Int. Conf. on Parallel Problem Solving from Nature (PPSN VIII). LNCS 3242, 21–30.

[2] Coello Coello, C. A., Van Veldhuizen, D. A., and Lamont, G. B. (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems.* Kluwer Academic Publishers, New York.

[3] Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2001). *Introduction to Algorithms.* 2nd Edition, McGraw Hill, New York.

[4] Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms.* Wiley, Chichester.

[5] Droste, S., Jansen, T., and Wegener, I. (2002). On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science 276, 51–81.

[6] Ehrgott, M. (2000). Approximation algorithms for combinatorial multicriteria optimization problems. Int. Transactions in Operational Research 7, 5–31.

[7] Giel, O. (2003). Expected runtimes of a simple multi-objective evolutionary algorithm. In Proc. of the 2003 Congress of Evolutionary Computation (CEC), 1918–1925.

[8] Giel, O. and Wegener, I. (2003). Evolutionary algorithms and the maximum matching problem. In Proc. of the 20th Ann. Symp. on Theoretical Aspects of Computer Science (STACS). LNCS 2607, 415–426.

[9] Hamacher, H. W. and Ruhe, G. (1994). On spanning tree problems with multiple objectives. Annals of Operations Research 52, 209–230.

[10] Laumanns, M., Thiele, L., Zitzler, F., Welzl, E., and Deb, K. (2002). Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. Proc. of the 7th Int. Conf. on Parallel Problems Solving from Nature (PPSN VII). LNCS 2439, 44–53.

[11] Neumann F. (2004). Expected run times of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. In Proc. of the 8th. Int. Conf. on Parallel Problem Solving from Nature (PPSN VIII). LNCS 3242, 80–89.

[12] Neumann, F. and Wegener, I. (2004). Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. In Proc. of Genetic and Evolutionary Computation Conference (GECCO 2004). LNCS 3102, 713–724.

[13] Raidl, G. R. and Julstrom, B. A. (2003). Edge sets:

an effective evolutionary coding of spanning trees. IEEE Trans. on Evolutionary Computation 7, 225–239.

[14] Scharnow, J., Tinnefeld, K., and Wegener, I. (2002). Fitness landscapes based on sorting and shortest paths problems. Proc. of the 7th Conf. on Parallel Problem Solving from Nature (PPSN VII). LNCS 2439, 54–63.

[15] Zhou, G. and Gen, M. (1999). Genetic algorithm approach on multi-criteria minimum spanning tree problem. European Journal of Operational Research 114, 141–152.

[16] Zitzler, E., Laumanns, M., Thiele, L., Fonseca, C. M., and Grunert da Fonseca, V. (2003). Performance assessment of multi-objective optimizers: An analysis and review. IEEE Trans. on Evolutionary Computation 7, 117–132.