Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Diploma Thesis

Stochastic Methods for Multi-Objective Optimization

Roland Erwin Kurmann

Summer 2001

Supervisors:
Dr. Eckart Zitzler
Prof. Dr. Lothar Thiele

This report was written with LaTeX 2_ε, KOMA-Script, Mathematica and xfig.

Abstract

Real-world optimization problems often involve multiple, competing objectives in a highly complex search space. Multi-objective problems distinguish themselves from single-objective problems in that when preference information is absent no optimal solution is clearly defined but rather a set of alternative trade-off solutions exist, which are called the Pareto-optimal front. Generating Pareto-optimal solutions in large, complex search spaces is usually intractable and cannot be solved exactly. Thus, efficient approximation methods, that are able to deal with multiple, competing objectives and large, complex search spaces are required.

Among the heuristic methods capable of dealing with large search spaces, the multi-objective evolutionary algorithms (MOEA) are assumed to be one of the most promising approaches. This is the case, because, thanks to their population-based structure, they are capable of generating a set of trade-off solutions in one optimization run. A lot of research has been conducted addressing the field of MOEAs and many methods have been developed. On the contrary alternative multi-objective methods, have not been studied in such depth and only few alternative approaches have been proposed. This is mainly due to the fact that generating a set of trade-off solutions seems to be more difficult using alternative optimization principles. For example local search based techniques, such as simulated annealing and tabu search, only generate one solution per run and a set of solutions are needed to describe the trade-off front.

The subject of this thesis was to find, classify, and compare experimentally alternative stochastic methods for multi-objective optimization and furthermore to relate the performance of alternative methods to that of MOEAs using multi-objective test problems. Emerging algorithmic improvements and combination possibilities should be investigated and experimentally verified.

Eight alternative multi-objective approaches based on simulated annealing and tabu search were found in literature. An approximation set of trade-off solutions is generated by these alternative approaches by performing several runs. A classification into three classes according to the handling of runs is proposed: independent runs, dependent parallel runs and dependent subsequent runs. In order to compare the approaches extensive computational experiments on the multi-objective 0/1 knapsack problem were performed. The class of methods performing dependent parallel runs attained the best results among the alternative methods. Alternative methods were able to find broader trade-off fronts among solutions in comparison with those generated by MOEAs. A comparison of alternative methods with MOEAs showed that alternative methods are able to find broader trade-off fronts. After further investigations this observation was attributed to the random walk phenomenon, where solutions in the center of trade-off fronts are favored. Several combination types of the tabu search optimization principle and the evolutionary algorithm were implemented and tested, trying to overcome the random walk phenomenon. Finally, an experimental weight-based multi-objective evolutionary algorithm, which avoided the random walk phenomenon was developed, tested and discussed.

Contents

1. Introduction

Many real-world optimization problems involve multiple, conflicting objectives in a highly complex search space. Usually these problems have been modeled as single-objective problems since many single-objective methods are available and well understood. But the reduction of conflicting objectives is equivalent to taking an a priori decision. In real-world problems there is usually only little knowledge available about the problem structure and a decision is therefore in many cases arbitrary. Therefore, it is desirable to postpone decisions to later stages of the optimization process. Modeling these real-world problems as multi-objective problems enables the possibility of a posteriori decisions.

In a multi-objective problem the multiple, conflicting objectives are retained in the model. Single-objective optimization problems result in single best solutions whereas multi-objective optimization problems give rise to a set of alternative trade-off solutions, the so called Pareto-optimal front. These Pareto-optimal solutions are optimal in a wider sense that no other solutions in the search space are superior to them when all objectives are considered. A decision can then be made on the basis of the alternative trade-off solutions considering higher level knowledge about the real-world problem.

As a result, methods are needed which are capable of finding the Pareto-optimal front. However, multi-objective problems which have highly complex search spaces are hard to solve exactly, since even most single-objective counterparts are intractable. Thus, heuristic methods are needed in order to approximate the Pareto-optimal fronts, among which multi-objective evolutionary algorithms (MOEA) are the most common methods. Evolutionary algorithms seem to be well suited for multi-objective problems, since they work with a population of solutions. Thanks to this they are able to generate a set of trade-off solutions in one optimization run. During the last decade many multi-objective evolutionary algorithms have been developed (Deb 2001), on the contrary only few alternative multi-objective methods based on other optimization principles, such as simulated annealing and tabu search, have been proposed (Sait and Youssef 1999). This is the case because it is not as clear how to generate an approximation of the Pareto-optimal front in one optimization run for alternative multi-objective methods as for MOEAs. Moreover, the supposition exists that multi-objective evolutionary algorithms are more appropriate for finding approximations of Pareto-optimal fronts than other search strategies (Fonseca and Fleming 1995).

Up to now experimental performance studies comparing different multi-objective approaches are restricted to the field of MOEAs (Zitzler and Thiele 1999), beside a single study (Tuyttens, Teghem, Fortemps, and Nieuwenhuyze 2000), which compares a simulated annealing approach with an exact two phase method on a bicriteria assignment problem. This situation led to the initial questions for this diploma thesis:

- Which alternative stochastic methods for multi-objective optimization have been proposed?
- What are the basic principles of existing alternative methods and how can the alternative approaches be classified?
- What is the performance of the alternative methods among each other and especially compared to multi-objective evolutionary algorithms?
- Can alternative or evolutionary multi-objective methods be improved by a combination of concepts?

The continuing chapters of this thesis are structured in the following manner: Chapter 2 introduces the key concepts of multi-objective optimization and defines the terminology used in this thesis. Chapter 3 introduces the basic principles of alternative methods and proposes a classification of alternative methods. A comparison of the alternative and the evolutionary multi-objective methods based on computational experiments is presented in Chapter 4. Chapter 5 is devoted to combinations of methods. Finally, conclusions and future perspectives are given in Chapter 6.

2. Definitions

Many real-word optimization problems in a highly complex search space can be modeled as multi-objective combinatorial optimization problems.

Definition 1 *A general multi-objective combinatorial problem (*MOCO*) with m dimensions (parameters) and k objectives can be stated as follows:*

$$\begin{array}{ll} \textit{“min”} & \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})) \\ \textit{s.t.} & \boldsymbol{x} \in S \end{array} \tag{2.1}$$

*and where S is a discrete, finite set of feasible solutions (*decision space*), $Z \subseteq \mathbb{R}^k$ is the* objective space*, and a set of k functions $\boldsymbol{f}(\boldsymbol{x})$: $S \to Z$. A point $\boldsymbol{x} \in S$ is called* decision vector *and a point $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x})$ is called* objective vector.

Note that if f_i is to be maximized, it is equivalent to minimize $-f_i$.

In the case of conflicting objectives it is not possible to find a single solution that would optimize all the objectives simultaneously. In multi-objective optimization, decision vectors are regarded as optimal if their components cannot be improved without deterioration to at least one of the other components. This is formalized by the concept of dominance.

Definition 2 *Let $\boldsymbol{a}, \boldsymbol{b} \in S$ be arbitrary decision vectors.*

1. *A decision vector $\boldsymbol{a}$ is said to* dominate *a decision vector $\boldsymbol{b}$ (also written as $\boldsymbol{a} \prec \boldsymbol{b}$) if and only if*

$$\begin{array}{ll} \forall i \in \{1, \ldots, k\}: & f_i(\boldsymbol{a}) \leqslant f_i(\boldsymbol{b}) \quad \wedge \\ \exists j \in \{1, \ldots, k\}: & f_j(\boldsymbol{a}) < f_j(\boldsymbol{b}). \end{array} \tag{2.2}$$

2. *A decision vector $\boldsymbol{a}$ is said to* strongly dominate *a decision vector $\boldsymbol{b}$ if and only if*

$$\forall i \in \{1, \ldots, k\}: \quad f_i(\boldsymbol{a}) < f_i(\boldsymbol{b}). \tag{2.3}$$

3. *In this study $\boldsymbol{a}$ is said to* cover *$\boldsymbol{b}$ ($\boldsymbol{a} \preceq \boldsymbol{b}$) if and only if $\boldsymbol{a} \prec \boldsymbol{b}$ or $\boldsymbol{f}(\boldsymbol{a}) = \boldsymbol{f}(\boldsymbol{b})$.*

The definitions for a maximization problem ($\succ, \succeq$) are analogical.

Based on the above relation, non-dominated and Pareto-optimal solutions can be defined:

Definition 3 *Let $\boldsymbol{a} \in S$ be an arbitrary decision vector.*

1. *The decision vector $\boldsymbol{a}$ is said to be* non-dominated *regarding a set $S' \subseteq S$ if and only if there is no vector in S' which dominates $\boldsymbol{a}$; formally*

$$\nexists \boldsymbol{a}' \in S' : \boldsymbol{a}' \prec \boldsymbol{a}. \tag{2.4}$$

   If it is clear within the context which set S' is meant, it is left out.

2. *The decision vector $\boldsymbol{a}$ is* Pareto-optimal *if and only if $\boldsymbol{a}$ is non-dominated regarding S. An objective vector is Pareto-optimal if the corresponding decision vector is Pareto-optimal.*

3. *The set S' is called* Pareto-optimal set *if and only if*

$$\forall \boldsymbol{a}' \in S' : \nexists \boldsymbol{a} \in S : \boldsymbol{a} \prec \boldsymbol{a}'. \tag{2.5}$$

   The corresponding set of objective vectors is denoted as Pareto-optimal front.

Note that a Pareto-optimal set does not necessarily contain all Pareto-optimal solutions.

Just locating one non-dominated solution is very often a NP-hard task and we will in practice have to settle for an approximation of the Pareto-optimal set.

Definition 4 *A set $A \subset S$ is called an* approximation set *regarding the Pareto-optimal set if and only if all decision vectors $\boldsymbol{a} \in A$ are pairwise non-dominated.*

In this study the approximation set is also referred as *archive* of non-dominated solutions.

Weights for scalarization functions are defined as follows:

Definition 5 *The $\boldsymbol{\lambda}$-vector space Λ is defined as:*

$$\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}^k | \lambda_i \in [0,1], \sum_{i=1}^{k} \lambda_i = 1\} \tag{2.6}$$

A normalized vector $\boldsymbol{\lambda} \in \Lambda$ is called weight.

3. Methods

Literature research has provided eight alternative approaches for multi-objective optimization based on neighborhood or local search based techniques, namely simulated annealing and tabu search. The basic concepts of these two optimization techniques will be introduced first, then the general adaptations of local search based techniques will be presented and finally a classification of the found methods will be given.

3.1. Single-Objective Methods

Simulated annealing and tabu search are well known among the general single-objective optimization methods. These so called meta-heuristics are widely used for solving practical problems. Much research has been done and many adaptations for specific problems were developed. Furthermore, advantages of these methods are their good performance, their general applicability and their simplicity (Sait and Youssef 1999). Thus, it is logical that these were taken for an extension to multi-objective problems.

Simulated annealing and tabu search are local search based methods, that means they work on one current solution. A solution is improved in one iteration considering neighboring solutions or short the neighborhood of that solution. The neighborhood is problem specific and must be defined in advance of an optimization. Furthermore, the neighborhood structure should connect the entire solution space. Neighboring solutions are obtained through one or a few changes to the current solution. Optimization proceeds by moving through the neighborhood structure. Because new neighbors may be worse than the current solution, these meta-heuristics are capable of getting over local optima. Finally, simulated annealing and tabu search rely on the possibility of generating feasible initial solutions.

The basic simulated annealing algorithm (Algorithm 1) proceeds by generation of a random neighbor per iteration. If the random neighbor is better than the current solution, it is accepted as the new current solution. However if the random neighbor is worse than the current solution, it is accepted with a probability < 1. The acceptance probability depends on the extent of the deterioration and a parameter called temperature. The temperature normally starts on a level that is high enough for the most neighbors to be accepted and is gradually lowered to accept only smaller and smaller deteriorations. The way the temperature is lowered is referred to as cooling scheme.

The basic tabu search algorithm (Algorithm 2) repeatedly moves from the current solution to the best solution of its neighborhood. In order to prevent returning to

[1] Note, an improvement is always accepted with probability 1.

[2] The function $argmin$ returns the "argument of the minimum". Here, the solution y with smallest objective function value f among the non-tabu neighbors is returned.

Algorithm 1 Basic simulated annealing

$x := initialSolution()$;
$bestX := x$;
$T := initialTemp()$;
repeat
  select a random solution $y \in neighborhood(x)$;
  if $P(x, y, T) > random[0, 1]$ **then**
    $x := y$;
  end if
  if $f(x) < f(bestX)$ **then**
    $bestX := x$;
  end if
  $update(T)$;
until stop condition is fulfilled

$P(x, y, T)$[1] $:= \min\left\{1, e^{\frac{f(x)-f(y)}{T}}\right\}$

Algorithm 2 Basic tabu search

$x := initialSolution()$;
$bestX := x$;
$tabulist := \emptyset$;
repeat
  $y := argmin$[2]$\{f(y)|y \in neighborhood(x) \wedge move(x, y) \notin tabulist\}$;
  if $length(tabulist) > maxTabuListLength$ **then**
    remove the oldest element from the $tabulist$;
  end if
  x := y;
  add $move(y, x)$ as the newest element to the $tabulist$;
  if $f(x) < f(bestX)$ **then**
    $bestX := x$;
  end if
until stop condition is fulfilled

already visited solutions, the reverse move is declared tabu. Moves that have been declared tabu are kept in a tabu list. The tabu moves are disregarded for a period of neighborhood moves, as defined by the length of the tabu list.

3.2. Multi-Objective Adaptations

In the literature various ways to adapt a single-objective meta-heuristic have been suggested. Several researchers have proposed adaptations and presented multi-objective algorithms based on simulated annealing (Serafini 1994; Ulungu, Teghem, Fortemps, and Tuyttens 1999; Engrand 1997; Czyzak and Jaszkiewicz 1998; Suppapitnarm, Seffen, Parks, and Clarkson 2000) and tabu search (Hansen 1997c; Gandibleux, Mezdaoui, and Fréville 1997). Most of the above work was developed independently and has similar or redundant parts. This section mentions general extensions for local search based methods, especially for simulated annealing and tabu search as reported in the above publications. In local search based methods solutions have to be compared with each other respecting the objective function. Since there are multiple objectives in a multi-objective environment, adaptations must at least address that.

A multi-objective simulated annealing method must adapt the probability for accepting neighboring solutions, denoted as acceptance probability. In the single-objective environment only two cases exist, the new solution is better or worse. By comparing two objective vectors $\boldsymbol{x}, \boldsymbol{y} \in S$ in MOCO three cases arise:

- $\boldsymbol{y}$ dominates or is equal to $\boldsymbol{x}$,
- $\boldsymbol{y}$ is dominated by $\boldsymbol{x}$, or
- $\boldsymbol{y}$ is non-dominated with respect to $\boldsymbol{x}$.

The first two cases can easily be reduced to the single-objective case. The third case can be seen either as a deterioration (weak acceptance) or as an improvement (strong acceptance). Several acceptance probabilities have been proposed (Serafini 1994; Ulungu et al. 1999; Engrand 1997; Suppapitnarm et al. 2000). Two characteristic acceptance probabilities are presented below.

First we discuss a weighted sum approach, where the differences on each objective are accumulated. The probability is defined by the expression:

$$P_{WS}(\boldsymbol{x}, \boldsymbol{y}, T, \boldsymbol{\lambda}) := \min \left\{ 1, e^{\sum_{j=1}^{k} \lambda_j (f_j(\boldsymbol{x}) - f_j(\boldsymbol{y}))/T} \right\} \tag{3.1}$$

where T is the temperature and $\boldsymbol{\lambda}$ is the weight vector. This probability represents the family of probabilities, where the objectives are aggregated in advance by a certain function $F(\boldsymbol{f}(\boldsymbol{x}))$. Such an aggregation reduces the multiple objectives to a single one, thus all vectors can be compared, even the non-dominated. Therefore, strong and weak acceptance coincide.

A strong acceptance probability, inspired by the Tchebycheff metric with the reference vector $\boldsymbol{x}$ is defined as follows:

$$P_{ST}(\boldsymbol{x}, \boldsymbol{y}, T, \boldsymbol{\lambda}) := \min \left\{ 1, \min_{j \in \{1,\ldots,k\}} \left\{ e^{\lambda_j (f_j(\boldsymbol{x}) - f_j(\boldsymbol{y}))/T} \right\} \right\} \tag{3.2}$$

Algorithm 3 $updateArchiv(\boldsymbol{x}, A)$

```
if x is non-dominated by all elements of A then
  for all dominated a ∈ A by x do
    A := A\a;
  end for
  A := A ∪ {x};
end if
```

This probability is a member of the Pareto-based family. A weak acceptance results, if the inner $\min()$ is replaced by $\max()$.

In order to find a "best" neighbor in a multi-objective tabu search algorithm, neighbors must be compared. The common approach is scalarization of objectives. Popular methods are the weighted sum

$$\boldsymbol{\lambda} \cdot \boldsymbol{f}(\boldsymbol{x}) \tag{3.3}$$

or the weighted Tchebycheff metric

$$\|\boldsymbol{z} - \boldsymbol{z}^{ideal}\|_{\infty}^{\boldsymbol{\lambda}} = \max_{i \in \{1,\ldots,k\}} \{\lambda_i | z_i - z_i^{ideal}|\} \tag{3.4}$$

where $\boldsymbol{z}^{ideal} \in \mathbb{R}^k$ is the *ideal objective vector*. The components z_i^{ideal} are obtained by minimizing each of the objective functions individually. Other functions than the weighted sum are used, because the weakness of weighting methods is that not all Pareto-optimal solutions can be found unless the problem is convex (Steuer 1986). Weighted sums in conjunction with multi-objective tabu search do not seem to be a practical problem, since every iteration a new sum is calculated.

Above adaptations contain weights, which may influence or even steer the search process. Thus, weights must be set carefully. Different weight setting mechanisms are presented in literature, these will be discussed in Section 3.3.

Since in a multi-objective environment not only one single solution exists, but a set of solutions, these solutions have to be stored. The common approach is to maintain an archive with a simple update procedure (Algorithm 3). Every new generated point is checked against the archive for non-dominance. If the new vector is non-dominated, it is inserted and all dominated vectors will be removed. This simple update procedure may run into problems, since there exist a huge number of non-dominated points. After a certain problem size, it is impossible to store all generated solutions. An appropriate reduction mechanism must be applied.

Many improvements and extensions for the single-objective methods have been proposed. In general these improvements can also be applied to multi-objective methods.

3.3. Classification

This section aims to classify the eight multi-objective approaches. A survey and classification for a wider scope of MOCO methods, especially exact procedures and dedicated heuristics can be found in (Ehrgott and Gandibleux 2000). Main properties and theoretical results of MOCO problems are also discussed.

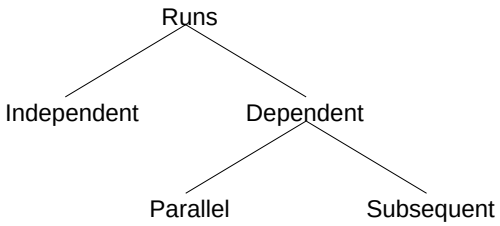


Figure 3.1.: Classification Schema

The local search based methods work on a single solution, the current solution. The current solution is modified in every iteration. A single start of such a procedure can only approximate a small region of the Pareto-optimal front. Several runs with different parameters (e.g. weights), starting from different initial solutions are necessary. Each run has its own current solution, adopted form the MOEA terminology, these current solutions will be denoted as population. Runs can be started independently, as several single-objective methods and the generated objective vectors can be merged to one trade-off front. Ideas have been developed where runs interact with each other. Interacting runs optimize the population simultaneously in each iteration while information is shared, whereas the approaches differ through the type of interaction.

We believe that the interaction of runs is an appropriate classification criterion. Then, the reported methods can be divided, as shown in Figure 3.1, into three classes: independent runs, dependent parallel runs and dependent subsequent runs. Independent runs do not have any interaction among each other. Dependent runs are distinguished by their type of interaction. Dependent parallel runs exchange information iteration by iteration, e.g. the current representation on the non-dominated front. Subsequent runs perform one run and restart using available information, e.g. the decision vectors from the archive. Each class with its methods is separately discussed in below sections.

3.3.1. Independent Runs

This class contains methods whose authors have mainly concentrated on other aspects than the interaction of runs, namely on the basic principles of adaptations as described in Section 3.2.

The general algorithm for independent runs is summarized in Algorithm 4. Initially, the number of runs must be determined. Starting from an empty archive, each run is started either in parallel or subsequently and optimizes its current solution. A run sets the scalarization weight $\boldsymbol{\lambda}$ at the beginning of a run. An optimization run is then started from a feasible initial solution. The current solution $\boldsymbol{x}_{t-1}$ is iteratively improved by the procedure $improveSolution()$ either using simulated annealing or tabu search techniques. The new obtained decision vector $\boldsymbol{x}_t$ is then processed by Algorithm 3. Weights may be periodically modified either using the information available during a single run which may be in our case the improvement of the objective vector from $\boldsymbol{x}_{t-1}$ to $\boldsymbol{x}_t$ or just randomly. This information may help guide the search through a better setting of $\boldsymbol{\lambda}$. At the end of the runs, all archives A_j are merged.

Algorithm 4 Independent Runs

```
r := numberOfRuns();
for all j ∈ {1, ..., r} do
  t := 0;
  λ := initialWeight();
  x_0 := initialSolution(S);
  A_j := ∅;
  repeat
    t := t + 1;
    x_t := improveSolution(x_{t-1}, λ);
    A_j := updateArchive(x_t, A_j);
    λ := updateWeight(λ, x_t, x_{t-1}, t);
  until stop condition is fulfilled
end for
A := mergeArchives(A_1, ..., A_r);
```

Serafini (1994), Ulungu et al. (1999) and Engrand (1997) independently started the adaptation of simulated annealing for multi-objective problems. They investigated adaptation techniques as described in Section 3.2. Engrand (1997) proposes a "return to base" option where the current solution is periodically replaced by a member of the archive A_j, aiming to exploit different regions of the Pareto-optimal front. Suppapitnarm et al. (2000) enhance Engrand's algorithm by implementing improvements known from single-objective simulated annealing. Suppapitnarm et al. (2000) also propose an archive-based solution acceptance where a solution $\boldsymbol{y}$ is always accepted if $\boldsymbol{y}$ is non-dominated regarding archive A_i or otherwise accepted with a probability < 1.

Gandibleux et al. (1997) propose a single run multi-objective tabu search algorithm performing weight updates. They suggest a modification of $\boldsymbol{\lambda}$ according to the level of improvement of $f_i(\boldsymbol{x}_t)$ compared to $f_i(\boldsymbol{x}_{t-1})$ in order to diversify the search to directions of "weak" improvements. In contrary to the general algorithm, not only the "best" neighboring solution $\boldsymbol{y}$ is considered for insertion into the archive but also further members of the neighborhood. The m next "best" neighbors are inserted where m is depending on the number of dominated solutions of A_i by $\boldsymbol{y}$. Gandibleux et al. (1997) innovates, additionally to the tabu list for moves, a weight update tabu list for each weight component λ_i in order to avoid too rapid changes of the same weight component λ_i. Since this method is only verified on a hand calculation example, it is not clear how good the approximation of the Pareto-optimal front for real-world problems actually is.

3.3.2. Dependent Parallel Runs

Czyzak and Jaszkiewicz (1998) proposed a simulated annealing algorithm using a population of solutions which interact at each iteration. Their approach is an adaptation of the population concept from evolutionary algorithms for multi-objective local search based methods. Weights are set through interaction. Population members influence each other aiming to spread the search to different regions of the Pareto-optimal front. The goal is to set the weights so that the points move away from the other population

Algorithm 5 Dependent Parallel Runs

```
t := 0;
X_0 := initialPopulation(S);
λ := initialWeight();
A := ∅;
repeat
  X_{t+1} := ∅;
  for all x ∈ X_t do
    λ := updateWeight(X_t, x, λ);
    y := improveSolution(x, λ);
    A := updateArchive(y, A);
    X_{t+1} := X_{t+1} ∪ {y};
  end for
  X_{t+1} := changePopulation(X_{t+1}, t);
  t := t + 1;
until stop conditions are fulfilled
```

members. Hansen (1998) has applied the population idea to tabu search.

The general principle is stated in Algorithm 5. Initially, a population of feasible initial decision vectors is generated by the procedure $initialPopulation()$. The weight $\boldsymbol{\lambda}$ is initialized as well, e.g. at random. Then the members of the population are "simultaneously" optimized in each iteration step. Two types of interaction exist. The first interaction of runs performs the procedure $updateWeight()$ in order to determine the search direction for the current decision vector $\boldsymbol{x} \in X_t$, the position of $\boldsymbol{f}(\boldsymbol{x})$ in the objective space is compared with the position of the other population members. The weight vector $\boldsymbol{\lambda}$ is then modified using this information. The procedure $improveSolution()$ optimizes the solution $\boldsymbol{x}$ either using simulated annealing or tabu search techniques. Afterwards, the archive is updated with the newly obtained vector $\boldsymbol{y}$ using Algorithm 3. A new population X_{t+1} is formed with the improved solutions $\boldsymbol{y}$. The second type of interaction is performed by the procedure $changePopulation()$. The population is periodically modified in order to control the search. A possible modification is doubling or deleting of members regarding each other's dominance.

Czyzak and Jaszkiewicz (1998) and Hansen (1998) suggest different procedures for $updateWeight()$. The procedure $updateWeight()$ of Czyzak and Jaszkiewicz (1998) determines the closest solution $\boldsymbol{x}' \in X_t$ in the objective space to $\boldsymbol{x}$, which is non-dominated with respect to $\boldsymbol{x}$. These two solutions are compared on each objective f_i and the weight component λ_j is changed:

$$\lambda_j := \begin{cases} \beta\lambda_j & \text{if } f_j(\boldsymbol{x}) \leqslant f_j(\boldsymbol{x}') \\ \lambda_j/\beta & \text{if } f_j(\boldsymbol{x}) > f_j(\boldsymbol{x}') \end{cases} \tag{3.5}$$

where $\beta < 1$ is a constant close to one (e.g. 0.95) and $\boldsymbol{\lambda}$ is normalized after the modification. This weight update aims to lead $\boldsymbol{x}$ away from the closest neighbor $\boldsymbol{x}'$. Further search is intensified in the direction where $\boldsymbol{x}$ is comparatively better than $\boldsymbol{x}'$.

The algorithm of Hansen (1998) determines a new search direction for each population member $\boldsymbol{x}'$ according to the population of solutions X_t. Instead of only considering the closest member of the population, all non-dominated members are considered.

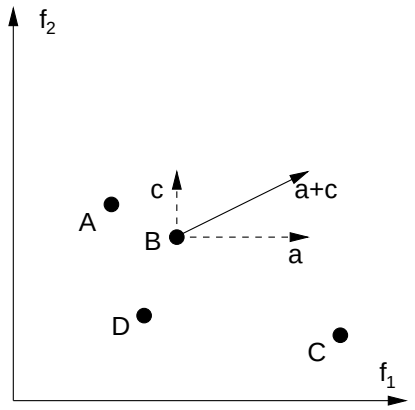


Figure 3.2.: Population-based direction setting for a maximization problem, where A, B, C and D are population members in the objective space and a and c are proximity values.

Algorithm 6 updateWeight($X_t, \boldsymbol{x}, \boldsymbol{\lambda}$): *Population-based weight setting*

$\boldsymbol{\lambda} := \mathbf{0}$;
for all $\boldsymbol{x}' \in X_t$ which are non-dominated by $\boldsymbol{x}$ **do**
  $w := g(d(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}')))$;
  for all objectives f_j where $f_j(\boldsymbol{x}) > f_j(\boldsymbol{x}')$ **do**
    $\lambda_j := \lambda_j + w$;
  end for
end for
if $\boldsymbol{\lambda} = \mathbf{0}$ **then**
  $\boldsymbol{\lambda} := randomWeight()$;
end if
$\boldsymbol{\lambda} := normalize(\boldsymbol{\lambda})$;

The $updateWeight()$ procedure for a maximization problem is given in Algorithm 6 where $d()$ is a distance function in the objective space, e.g. the Manhattan distance norm $d(\boldsymbol{z}, \boldsymbol{z}') = \sum_{j=1}^{k} |z_j - z'_j|$ and $g()$ is a proximity function, e.g. $g(d) = 1/d$. The proximity value w is calculated for all non-dominated population members. A comparison on each objective is performed, comparatively better objective directions are maintained. The closer a point, the more influence it gains. An example for a maximization problem is shown in Figure 3.2. For the procedure $changePopulation()$ Hansen proposes the copying of a member, where a random member is doubled, while another random member is deleted. The hope is that new search directions can be found. He also describes a more complex $changePopulation()$ procedure where the population size is dynamically changed using a certain domination degree among the population members. This can be used if the population size cannot be determined in advance. Additional publications about multi-objective optimization using tabu search are available. Hansen (1997a) describes a modified version suggesting the use of hash functions in order to yield performance improvements. The effect of different scalarization functions is studied in (Hansen 2000).

Algorithm 7 Dependent Subsequent Runs (CHESS)

$A := updateArchive(initialSolution(S), \emptyset)$;
repeat
  $D(A, \boldsymbol{x}) := createObjectiveFunction(A)$; *(*single objective*)*
  $\boldsymbol{x} :=$ **call** $singleObjectiveMethod(D(), A)$;
  $A := updateArchive(\boldsymbol{x}, A)$;
until stop conditions are fulfilled

A few applications of the above methods are reported in literature. Hansen (1997b) adapts his method for the multi-objective knapsack problem. Jaszkiewicz (1997) uses the parallel simulated annealing method for a multi-objective nurse scheduling problem. Czyzak and Jaszkiewicz (1997) apply parallel simulated annealing to complex manufacturing systems. Viana and Sousa (2000) apply both methods presented above to a multi-objective resource constraint project scheduling problem.

3.3.3. Dependent Subsequent Runs

In the third class of methods, several runs are performed subsequently. Information from previous runs, e.g. the archive, is used to control the search. Borges (2000) presented the method CHESS (Changing Horizon Efficient Set Search) which subsequently calls single-objective procedures using a specific objective function providing a Pareto-based objective function. Since to our knowledge CHESS is the only representative of this class, this algorithm will be described here.

CHESS, described in Algorithm 7, does not extend a single-objective method, but it makes use of them. A single-objective method is called providing a certain objective function $D()$ which relates a point $\boldsymbol{x}$ with the archive A:

$$D(A, \boldsymbol{x}) = \begin{cases} \text{non-negative}, & \text{if } \boldsymbol{x} \text{ is dominated by a point in } A, \\ \text{negative}, & \text{otherwise.} \end{cases} \tag{3.6}$$

The single-objective method minimizes objective $D()$. Thus, obtaining negative values is equivalent to finding non-dominated points. The objective function $D()$ can be seen as a distance measure of the point $\boldsymbol{x}$ to the archive A. Many candidates for $D()$ are possible, a relationship of $\boldsymbol{x} \in S$ with A based on the dominance concept is suggested in the publication. First, the distance of two vectors $\boldsymbol{x} \in S$ and $\boldsymbol{a} \in A$ is defined:

$$d(\boldsymbol{a}, \boldsymbol{x}) = \min_{i \in \{1,\ldots,k\}} (f_i(\boldsymbol{x}) - f_i(\boldsymbol{a})) \tag{3.7}$$

According to the value d takes, three cases can be distinguished:

$$\begin{aligned} &\text{i. } d(\boldsymbol{a}, \boldsymbol{x}) > 0 \iff \boldsymbol{a} \text{ strongly dominates } \boldsymbol{x}. \\ &\text{ii. } d(\boldsymbol{a}, \boldsymbol{x}) < 0 \iff \boldsymbol{a} \text{ does not dominate } \boldsymbol{x}. \\ &\text{iii. } d(\boldsymbol{a}, \boldsymbol{x}) = 0 \iff \boldsymbol{a} = \boldsymbol{x} \text{ or } \boldsymbol{a} \text{ weakly dominates } \boldsymbol{x}. \end{aligned} \tag{3.8}$$

The distance of a point $\boldsymbol{x} \in S$ to the archive A follows:

$$D(A, \boldsymbol{x}) = \max_{\boldsymbol{a} \in A}(d(\boldsymbol{a}, \boldsymbol{x})) = \max_{\boldsymbol{a} \in A} \left(\min_{i \in \{1,\ldots,k\}} (f_i(\boldsymbol{x}) - f_i(\boldsymbol{a})) \right) \tag{3.9}$$

where

$$\begin{aligned} &\text{i. } D(A, \boldsymbol{x}) > 0 \iff \boldsymbol{x} \text{ is dominated.} \\ &\text{ii. } D(A, \boldsymbol{x}) < 0 \iff \boldsymbol{x} \text{ is non-dominated.} \\ &\text{iii. } D(A, \boldsymbol{x}) = 0 \iff \boldsymbol{x} \text{ is weakly-dominated or belongs to } A. \end{aligned} \tag{3.10}$$

Since A is updated during the optimization process, $D()$ differs for subsequent calls. Note, the objective function $D()$ does not incorporate any weights in contrary to the scalarization functions used by the approaches of the classes of independent and parallel runs.

The method CHESS is inspired by a simple idea. Assuming an *exact* single-objective method, subsequent single-objective calls using $D()$ return negative values, as long as non-dominated points are available. If a positive value is returned, no more non-dominated points are available and the entire Pareto-optimal front has been found, thus the method can be stopped.

4. Comparison of Methods

Alternative methods and multi-objective evolutionary algorithms will be compared with each other in this chapter. The comparison of the methods is based on experiments on multi-objective 0/1 knapsack problems. The Table 4.1 gives a survey of the implemented and tested methods. Concerning the alternative methods, representatives of each class as given in Section 3.3 were chosen. SPEA (Zitzler and Thiele 1999) and SPEA2 (Zitzler, Laumanns, and Thiele 2001) were taken as representatives for the multi-objective evolutionary algorithms, because a good performance is reported in the publications and their implementation was available for this study. Note, the method CHESS was only used in conjunction with tabu search, since Borges (2000) reported bad results for simulated annealing. This comparison was done in analogy to the study of (Zitzler and Thiele 1999); the same test problems and the same metrics were used.

Class	Optimization Principle	Method Name
Independent Runs	Tabu Search	ITS
	Simulated Annealing	ISA
Dependent Parallel Runs	Tabu Search	PTS
	Simulated Annealing	PSA[1]
Dependent Subsequent Runs	Tabu Search	CHESS
Multi-Objective Evolutionary Algorithms		SPEA
		SPEA2

Table 4.1.: Methods for Comparison

4.1. Test Environment

Test Problem

The multi-objective 0/1 knapsack problem as defined in (Zitzler and Thiele 1999) was chosen as MOCO test problem. This problem is an extension of the single-objective 0/1 knapsack problem. The single-objective problem consists of a set of items, weight and profit associated with each item, and an upper bound for the capacity of the knapsack. The goal is to find a subset of items which maximize the profit sum while not exceeding the capacity. A multi-objective problem results if k knapsacks are taken.

[1] PSA stands for parallel simulated annealing and not for Pareto Simulated Annealing (Czyzak and Jaszkiewicz 1998). The implementation differs as well.

The multi-objective 0/1 knapsack problem with m items and k objectives is formally defined as

$$\begin{aligned} \max \quad & f_i(\boldsymbol{x}) = \sum_{j=1}^{m} p_{ij} x_j \qquad i = 1, \ldots, k \\ \text{s.t.} \quad & \sum_{j=1}^{m} w_{ij} x_j \leqslant c_i \end{aligned} \tag{4.1}$$

where $\boldsymbol{x} = (x_1, \ldots, x_m) \in \{0,1\}^m$ is the decision vector, $x_j = 1$ if and only if item j is selected, p_{ij} is the profit of item j according to knapsack i, w_{ij} is the weight of item j according to knapsack i and c_i is the capacity of knapsack i.

Note, a slightly different multi-objective knapsack problem is defined in (Hansen 1997b; Czyzak and Jaszkiewicz 1998; Ulungu et al. 1999). They use one knapsack with k profit objectives.

The data set of (Zitzler and Thiele 1999) was used for the computational experiments. The following three test problems were used: 750 items with 2, 3 and 4 objectives. Uncorrelated profits and weights were chosen, where p_{ij} and w_{ij} are random integers in the interval [10, 100]. The capacity of knapsack i was set to

$$c_i = \frac{1}{2} \sum_{j=1}^{m} w_{ij}. \tag{4.2}$$

Implementation

The alternative methods of Table 4.1 were implemented for the multi-objective 0/1 knapsack problem. The representatives of the class of *independent runs* were implemented as shown in Algorithm 4 where no weight updates ($updateWeight()$) are performed. Thus, the weight $\boldsymbol{\lambda}$ is not changed during any run. The methods of the class of *dependent parallel runs* were implemented as given in Algorithm 5. New weights are calculated in every iteration by Algorithm 6. The procedure $changePopulation()$ was not used in experiments, thus the population size remained constant during optimization. *CHESS* has been implemented as shown in Algorithm 7 using the distance function $D()$ as given in Equation 3.9.

An archive as described in Algorithm 3 is maintained by all methods. The procedure $updateArchive()$ is called for every new generated solution by the subroutine $improveSolution()$. CHESS additionally calls the procedure $updateArchive()$ after every single-objective iteration.

The following neighborhood function for the multi-objective 0/1 knapsack problem was used:

1. Repeatedly remove one randomly selected (non-tabu) item until there is free space for all items outside, regarding all knapsacks.

2. Repeatedly insert one randomly non-selected (non-tabu) item until no more items can be inserted, regarding all knapsacks.

"Non-tabu" refers to the context of tabu search.

An initial solution was built by a modified version of the above neighborhood function:

1. Empty all knapsacks.

2. Repeatedly insert one randomly non-selected item until no more items can be inserted, regarding all knapsacks.

The "best" neighbor in tabu search is evaluated by the weighted sum as shown in Equation 3.3. Every run maintains its own tabu list, where the first item, which is inserted into the knapsack by the neighborhood function, is set tabu. This item cannot be removed the next l iterations, where l is the length of the tabu list. Thus, cycling to equal knapsack fillings are prevented for the next l iterations. As acceptance probability for simulated annealing the strong Tchebycheff-based probability as given in Equation 3.2 was chosen. The temperature T is annealed after every A_{step} iterations using $T := \alpha T$.

In order to obtain as general results for MOCO as possible, no special improvements for the multi-objective knapsack problem were considered.

Existing implementations of SPEA and SPEA2 for the multi-objective knapsack problem were used. Their implementation is described in (Zitzler and Thiele 1999; Zitzler et al. 2001). As an exception, the initial populations are not created as described in the papers, but with the above stated procedure, in order to enable a fair comparison, having equal start conditions for all methods.

Performance Metrics

The results of computational experiments will be compared using the $\mathcal{S}$ and $\mathcal{C}$ metrics of (Zitzler and Thiele 1999). These metrics are scaling-independent, thus the objectives do not have to be scaled for measuring. The metrics are cited below.

Definition 6 (Size of the space covered) *Let $X' = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l) \subseteq S$ be a set of l decision vectors. The function $\mathcal{S}(X')$ gives the volume enclosed by the union of the polytopes $p_1, \ldots, p_l$, where each p_i is formed by the intersections of the following hyperplanes arising out of $\boldsymbol{x}_i$, along with the axes: for each axis in the objective space, there exists a hyperplane perpendicular to the axis and passing through the point $(f_1(\boldsymbol{x}_i), \ldots, f_k(\boldsymbol{x}_i))$. In the two-dimensional case, each p_i represents a rectangle defined by the points $(0, 0)$ and $(f_1(\boldsymbol{x}_i), f_2(\boldsymbol{x}_i))$.*

Definition 7 (Coverage of two sets) *Let $X', X'' \subseteq S$ be two sets of decision vectors. The function $\mathcal{C}$ maps the ordered pair (X', X'') to the interval $[0, 1]$:*

$$\mathcal{C}(X', X'') := \frac{|\{\boldsymbol{a}'' \in X''; \exists \boldsymbol{a}' \in X' : \boldsymbol{a}' \succeq \boldsymbol{a}''\}|}{|X''|} \tag{4.3}$$

The value $\mathcal{C}(X', X'') = 1$ means that all points in X'' are dominated by or equal to points in X'. The opposite, $\mathcal{C}(X', X'') = 0$, represents the situation when none of the points in X'' are covered by the set X'. Note that both $\mathcal{C}(X', X'')$ and $\mathcal{C}(X'', X')$ have to be considered, since $\mathcal{C}(X', X'')$ is not necessarily equal to $\mathcal{C}(X'', X')$ (e.g., if X' dominates X'' then $\mathcal{C}(X', X'') = 1$ and $\mathcal{C}(X'', X') = 0$).

The size of the covered space obtained by metric $\mathcal{S}$ is also referred to as *area* in this study.

Methodology

In order to compare the methods, a common stop condition had to be defined. We decided to stop the optimization process after a certain number of objective function $\boldsymbol{f}$ evaluations. The advantage of this condition is the implementation independence. Moreover, the evaluation of the objective functions may be the most "expensive" operation in real-world problems and thus the most important regarding the run-time. The chosen numbers of evaluations for the different test problems are shown in Table 4.2:

	2 Objectives	3 Objectives	4 Objectives
Evaluations	500000	1000000	1500000

Table 4.2.: The number of objective function $\boldsymbol{f}$ evaluations for the 750 item knapsack problems

Per test problem 30 repetitions of each method were considered, where the random number seed was controlled. Therefore the initial population was the same in every repetition i, when an equal population size was used by the methods.

The archives obtained by the methods were compared using the $\mathcal{S}$ and $\mathcal{C}$ metric. Each repetition was measured by the metric $\mathcal{S}$, a sample of 30 $\mathcal{S}$ values was obtained per method. For each ordered pair of methods there was a sample of 30 $\mathcal{C}$ values per test problem according to the 30 repetitions.

In order to visualize the distributions of the samples, box plots are used. A box plot used in this study consists of two boxes, an inner and an outer one. The inner box summarizes 50% of the data and the outer summarizes the standard deviation of the distribution. The upper and lower ends of the inner box are the 75% and 25% quantiles, while the thick line within the boxes encodes the median. The upper and lower ends of the outer box are the arithmetic mean $\pm$ standard deviation. The scattered dots display the sample of values.

Parameter Settings

Since the methods, which are considered for the comparison, are quite different, the parameters had to be chosen with care in order to obtain sound results. Many preliminary tests on the two objective knapsack problem applying above methodology were performed in order to find adequate parameter settings. The parameter values which achieved the best results were chosen. Note, there is a trade-off for the tabu search based methods among the parameters the number of runs, neighborhood size and number of iterations. Trade-off parameters were examined by doubling and halving. The simulated annealing parameters of (Czyzak and Jaszkiewicz 1998) were taken as reference and adjusted to our knapsack problem instances. The parameters for the multi-objective evolutionary algorithms were taken from (Zitzler, Laumanns, and Thiele 2001). The parameters shown in Table 4.3 and 4.4 were chosen for the comparison of methods. The parameters marked with an asterisk (*) were preliminary tested.

Optimization Principle	Parameter	Value
Tabu Search	Neighborhood size	400*
	Tabulist Length l	3*
Simulated Annealing	Initial temperature T_0	120
	Initial acceptance probability P_0	>0.8
	Cooling factor α	0.9
	Annealing schedule A_{step}	2500
Multi-Objective Evolutionary Algorithms	Crossover rate p_c (one-point)	0.8
	Mutation rate p_m (per bit)	0.006
	Tournament size	2

Table 4.3.: Optimization principle specific parameters

Objectives	Parameter	Value
Independent & Dependent Parallel Runs		
2	Runs/Population Size	5*
	Iterations	250*
3	Runs/Population Size	10
	Iterations	250
4	Runs/Population Size	15
	Iterations	250
Dependent Subsequent Runs		
2	Single-Objective Iterations	800*
	Runs	1.6*
3	Single-Objective Iterations	800
	Runs	3.1
4	Single-Objective Iterations	800
	Runs	6.3
Multi-Objective Evolutionary Algorithms		
2	Population	250
	Elitist Archive	250
	Generations	2000
3	Population	500
	Elitist Archive	500
	Generations	2000
4	Population	750
	Elitist Archive	750
	Generations	2000

Table 4.4.: Problem dependent parameters per class

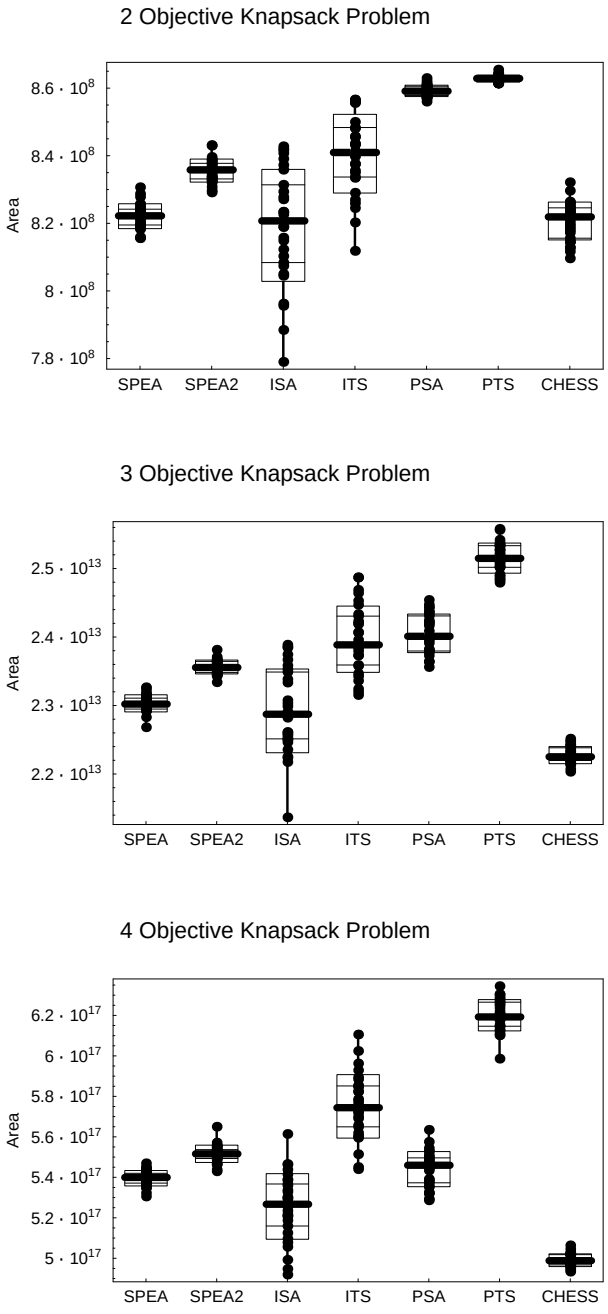


Figure 4.1.: Size of dominated space: Distributions of $\mathcal{S}$ values

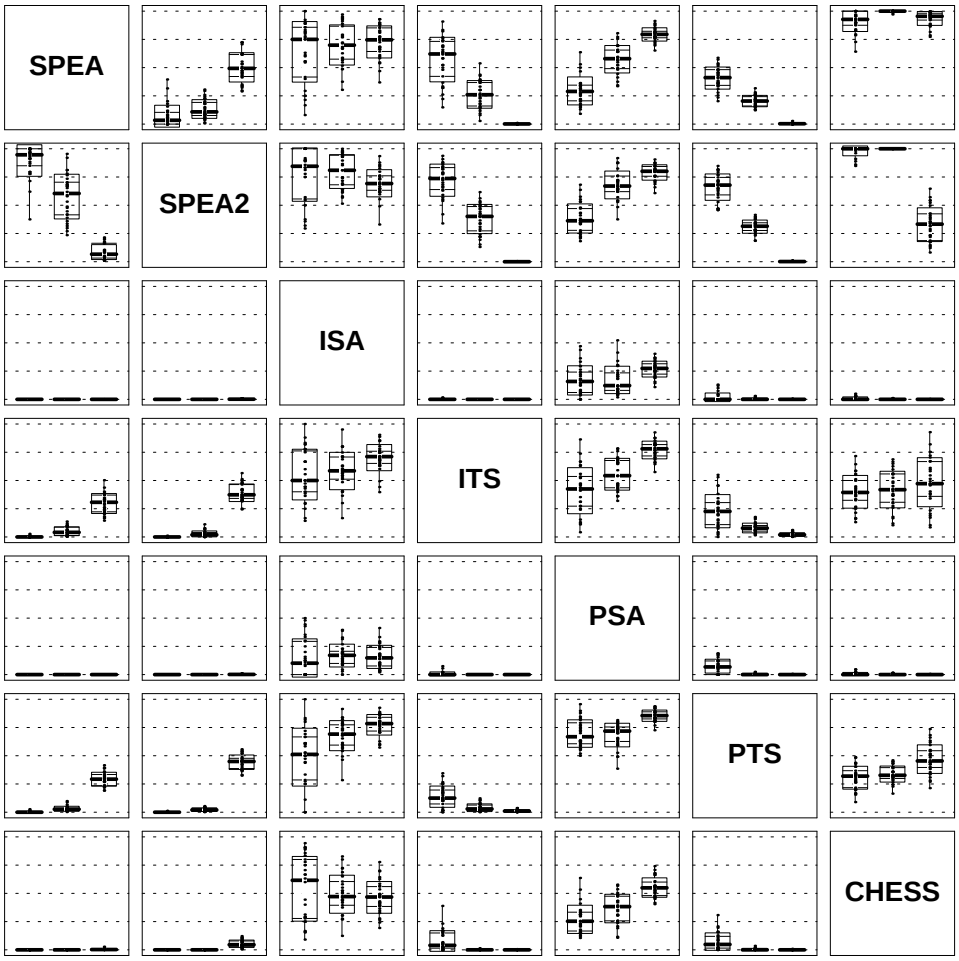


Figure 4.2.: Box plots based on the $\mathcal{C}$ metric. Each rectangle contains three box plots representing the distribution of the $\mathcal{C}$ values. The three box plots are related to the 2, 3 and 4 objective knapsack problems.

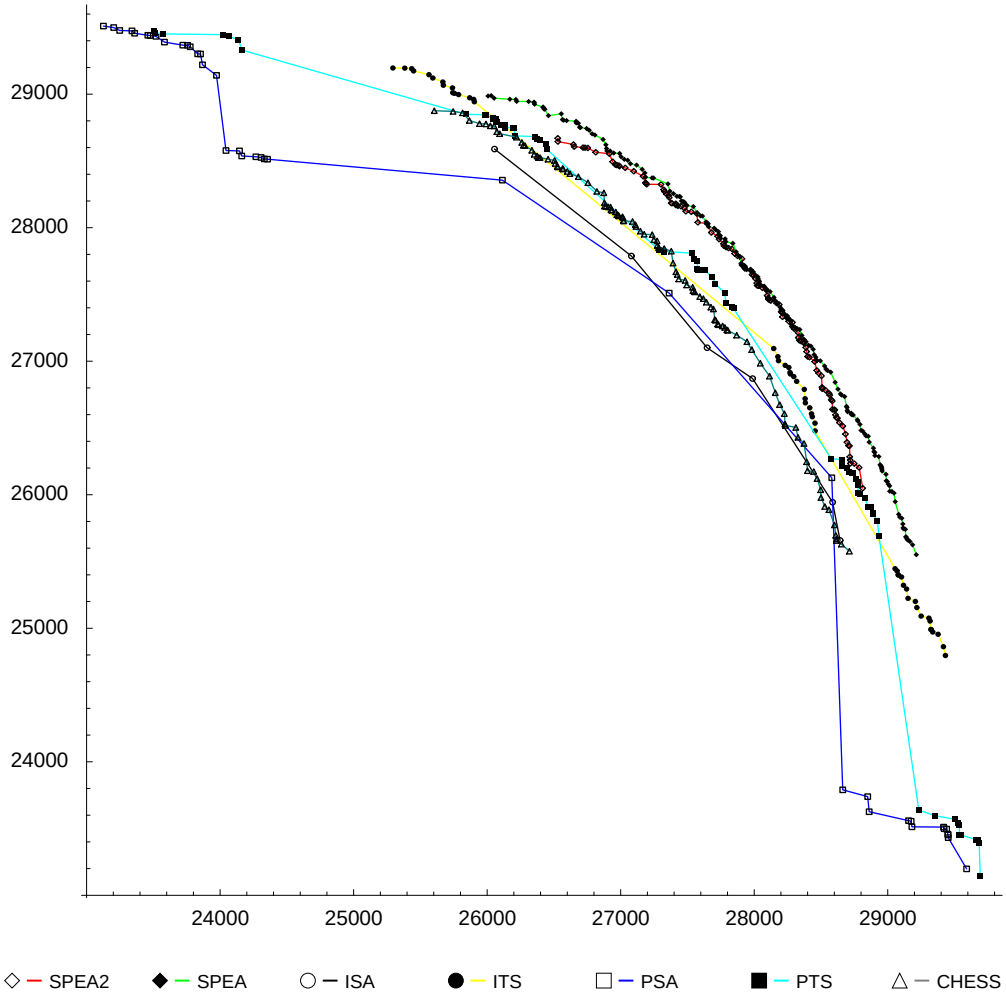


Figure 4.3.: Trade-off fronts for a two objective knapsack problem instance.

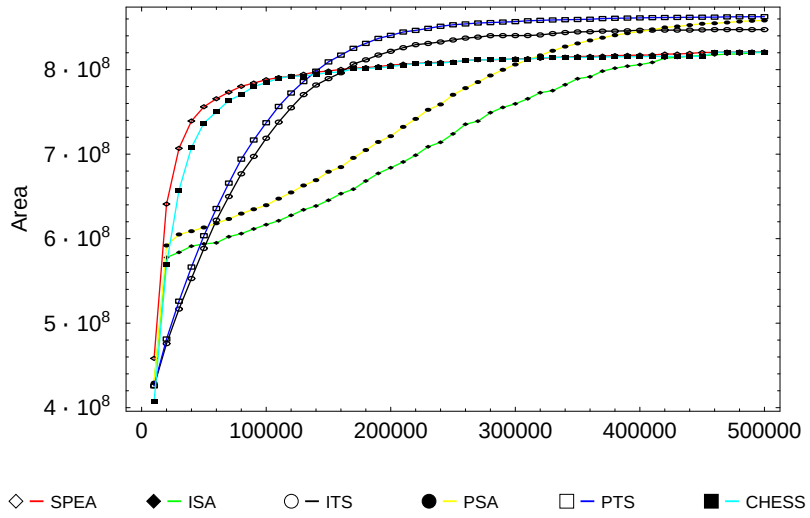


Figure 4.4.: In the graphs the development of the $\mathcal{S}$ metric median (area) achieved by the methods[3] is plotted over the number of objective function $\boldsymbol{f}$ evaluations. The approximation set snapshots were taken every 10000th evaluation of the objective function $\boldsymbol{f}$.

4.2. Results

Comparison results concerning the metric $\mathcal{S}$ (size of the covered space, also referred to as area) are depicted in Figure 4.1 and 4.4. The final distributions of $\mathcal{S}$ are shown in Figure 4.1, whereas the dynamic behavior over time of the methods is plotted in Figure 4.4. Figure 4.2 gives the direct comparison results of the $\mathcal{C}$ metric. In Figure 4.3 the trade-off fronts obtained by the methods of one repetition are shown in order to get a visual impression of the differences among the trade-off fronts generated by the methods. To show the differences clearly, a characteristic repetition has been chosen. Generally, visual inspections of non-dominated fronts have proved to be helpful for interpreting results obtained by the metrics. The following discussion of results is splitted into four parts, the first compares results of the alternative methods, the second describes the outcome of additional investigating experiments concerning alternative methods, the third is devoted to a comparison of the simulated annealing and the tabu search optimization principles and finally, the fourth discusses the experimental results obtained by the multi-objective evolutionary algorithms in comparison with those of the alternative methods.

Starting with the alternative approaches, the methods of the classes of dependent parallel runs and independent runs show similar results, where those of the CHESS method performing subsequent runs are inferior. Concerning the classes of independent and dependent parallel runs, the methods of the class of parallel runs achieve better results than those of independent runs, firstly, a better convergence is shown in Figure 4.1 and secondly, a bigger area is reported in Figure 4.2. Conspicuous is the

[3] SPEA2 could not be included in the comparison, because the output format of SPEA2 was not compatible with that of the other methods.

variance of the $\mathcal{S}$ metric distributions for the methods ISA and ITS, shown in Figure 4.1. The higher variance of these methods, which execute independent runs, can be attributed to the random weights which are initially set and not changed during the whole optimization. The performance comparison over time (cf. Figure 4.4) shows a similar convergence behavior for the methods of the classes of dependent parallel and independent runs, beside the differences already seen by the final distributions of $\mathcal{S}$ and $\mathcal{C}$ metrics. Since the methods of the class of dependent parallel runs yield a better result by metric $\mathcal{S}$ than the methods of the class of independent runs having random directions, it can be concluded that the population-based weight setting (Algorithm 6) is able to direct the population members to distinct regions of the Pareto-optimal front. Besides this the methods of the class of dependent parallel runs also achieve a broader trade-off front for the two objective case than the methods of the class of independent runs, as can be seen in Figure 4.3. A comparison of CHESS, the representative of the class of subsequent runs, shows that the generated trade-off fronts are not as broad as those obtained by the other methods (cf. Figures 4.3 and 4.1). Metric $\mathcal{C}$ also reveals the worst convergence to the Pareto-optimal front for CHESS. A different dynamic behavior of CHESS compared with the other methods is shown in Figure 4.4, CHESS converges fast in an early phase, but stagnates soon afterwards. Finally, a visual inspection of the approximation sets shows (cf. Figure 4.3) that methods of independent and dependent parallel runs pursue directions to separate regions of the non-dominated front, i.e., holes between these regions are recognizable. It can be summarized, that the methods of the class of dependent parallel runs show the best results among the alternative methods.

Additional experiments on the two objective knapsack problem investigating adaptive weight setting and alleviation of holes were performed. First, a modified version of the method ITS, denoted as ITSDIR, with preset uniform weights, representing optimal optimization directions was executed in order to verify the population-based weight setting of PTS method. The results obtained by ITSDIR (cf. Figure 5.3 in Section 5.2) were equivalent to those obtained by the method PTS, thus population-based weight setting is capable of finding the optimal weights for the two objective knapsack problem. Secondly, the number of runs performed by methods ITS and PTS were varied in order to remedy the holes. Generally, it can be said, the bigger the population, the smaller the holes. However, the convergence gets worse, since there is a trade-off between the population size and the number of iterations, while the number of objective function $\boldsymbol{f}$ evaluations remains fixed. Thirdly, another way for the alleviation of holes in method ITS was considered, the periodicial resetting of the weights, where every time a new random weight was set. The following effects with periodically changed weight vectors could be seen, the more frequent the weights were modified, (i) the smaller the holes, and additionally (ii) the narrower the entire trade-off front became. The Figure 5.3 in Section 5.2 shows the most extreme case of weight resetting, where a new random weight is set every iteration (method ITS1).

A comparison of the underlying optimization principles shows, according to the metrics $\mathcal{S}$ and $\mathcal{C}$, that the tabu search based methods outperform the simulated annealing based methods. First, the methods using the tabu search optimization technique achieve a larger area on the 2, 3 and 4 objective knapsack problems and secondly, the tabu search based methods are able to cover the simulated annealing based methods with at least 50% besides the exception of the methods ITS and PSA in the 2

objective case, whereas the coverage rates of the simulated annealing based methods mostly remain at 0%. A different convergence behavior to the Pareto-optimal front for the methods based on two different optimization principles is also revealed in Figure 4.4. Overall, the convergence velocity of the simulated annealing based methods is significantly slower than that of the tabu search based methods. Additionally, an inspection of Figure 4.3 shows that methods using simulated annealing generate less non-dominated points in a single run than the tabu search. Finally, another observation is that the population-based weight setting, in conjunction with simulated annealing, is less effective in higher objective dimensions than with tabu search, according to metric $\mathcal{S}$.

The multi-objective evolutionary algorithms SPEA and SPEA2 show a good convergence to the Pareto-optimal front for the two objective knapsack problem (cf. Figures 4.2 and 4.3), whereas the non-dominated fronts are narrower than those of the alternative methods PTS and PSA. On the two objective knapsack problem no trade-off fronts found by the other methods are able to dominate or cover solutions found by SPEA or SPEA2. However, this good convergence, compared to that of the other tested methods cannot be retained in higher objective dimensions. Fronts produced by SPEA2 (similar for SPEA) for four objectives are covered with almost 50% by ITS and PTS according to metric $\mathcal{C}$, whereas no solution generated by SPEA2 is able to cover any solution obtained by ITS or PTS. Concerning the dynamic behavior, the same type of convergence as that of CHESS can be seen in Figure 4.4, a rapid convergence in a first phase and stagnation afterwards.

Concludingly, two groups of methods can be identified either generating narrow or broad trade-off fronts. The methods SPEA, SPEA2 and CHESS can be assigned to the first group, generating narrow trade-off fronts and the methods ISA, ITS, PSA and PTS belongs to the second group, having broad trade-off fronts. A difference of these two groups is the guidance to the Pareto-optimal front, the first group controls the optimization process upon Pareto-dominance criteria, whereas the second group uses a weight-based scalarization function. Thus, the methods of the first group can be referred to as Pareto-based and the second as weight-based. Additionally, another difference between the two groups can be found, i.e., the Pareto-based methods converge very fast in a first phase, where the weight-based methods show a slower convergence velocity.

5. Combination of Methods

Results of chapter 4 have shown that the PTS method is able to generate broader fronts than the multi-objective evolutionary algorithms SPEA and SPEA2. Thus, extremer trade-offs could be generated by PTS. The question arises: Why are the trade-off fronts obtained by multi-objective evolutionary algorithms under consideration narrower? This question was tackled by modifying of methods and by performing further computational experiments.

5.1. Neighborhood

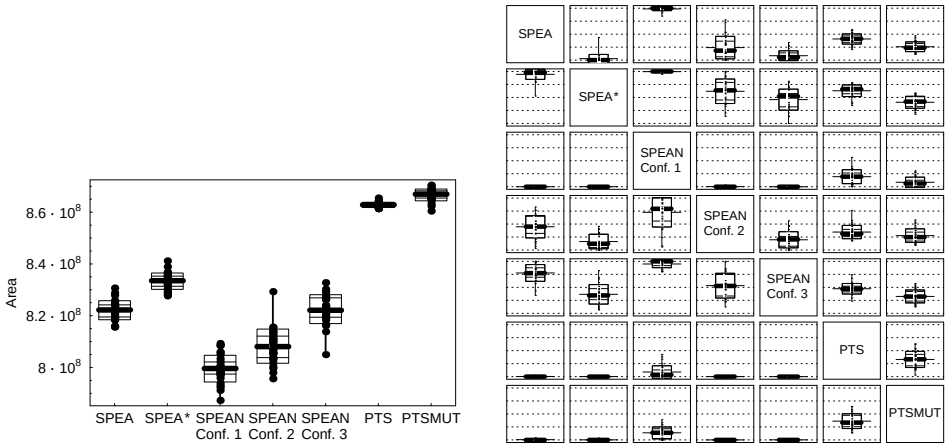
First the neighborhood concept of tabu search was examined and compared with that of evolutionary algorithms. The neighborhood concept describes the way how a decision vector is modified in order to obtain a new solution. The tabu search neighborhood concept consists of a neighborhood function which describes the modification of a given decision vector and the neighborhood. The neighborhood denotes the various neighboring solutions which are generated by the neighborhood function and evaluated by the weighted sum in order to choose the "best" neighbor. Evolutionary algorithms use mutation as a neighborhood function, where only one neighbor is considered. Mutation in contrary to the tabu search neighborhood function may produce decision vectors representing knapsacks which exceed knapsack capacity constraints. To approach the following questions, computational experiments have been performed on the two objective knapsack problem:

- Can evolutionary algorithms be improved by using the tabu search neighborhood function? Is the allowance of infeasible knapsacks a drawback for mutation? Is there a deterioration of PTS when using mutation?
- Can an improvement of evolutionary algorithms be achieved by using a tabu search neighborhood?

Computational experiments were made with modified versions of SPEA[1] and PTS. Methods were in a straight forward way combined since both source code bases could be used. Mutation was replaced in SPEA by the tabu search neighborhood function, the resulting method was called SPEA*. A version of ITS, denoted as PTSMUT, was implemented where the common neighborhood function is used instead of mutation. All neighbors resulting from mutation are repaired in order to obtain feasible knapsacks respecting the capacity constraints. The repairing mechanism of SPEA is used (Zitzler and Thiele 1999)[2]. The repairing procedure checks every item to find out if

[1] SPEA was chosen for improvements because the source code was available.

[2] The description of the repairing mechanism given in this study differs slightly from (Zitzler and Thiele 1999), since that of (Zitzler and Thiele 1999) does not represent the implementation exactly.



(a) (b)

Figure 5.1.: Distributions of metric $\mathcal{S}$ and $\mathcal{S}$ values of the neighborhood experiments.

there is still enough space, stopping by the first item which exceeds a capacity constraint. The order in which the items are checked, is determined by the maximum profit/weight ratio per item. The ratio q_j for item j is defined by

$$q_j = \max_{i=1}^{k} \left\{ \frac{p_{ij}}{w_{ij}} \right\}. \tag{5.1}$$

The items are considered in decreasing order of the q_j, thus items achieving the highest profit per weight are checked first. In order to experiment on evolutionary algorithms and neighborhood sizes, the tabu search neighborhood, using mutation as neighborhood function, was introduced in SPEA, the obtained method is called SPEAN. SPEAN evaluates the "best" neighbor by a weighted sum. The weights of every population member were initially set to random values.

Results are depicted in Figure 5.1. SPEA* shows an improvement over SPEA. The tabu search neighborhood function improves the convergence to the Pareto-optimal front, since a domination of 95% of SPEA* over SPEA is reported by metric $\mathcal{C}$. The bigger area (metric $\mathcal{S}$) obtained by SPEA* can be ascribed to the better convergence and a slightly broader trade-off front which could be seen by visual comparison of two trade-off fronts (cf. Figure 5.2). Since the effect of different mutation rates is not described in (Zitzler and Thiele 1999; Zitzler et al. 2001) and experiments on mutation rates were out of the scope of this work, it cannot finally be said if the better convergence is only corresponding to a more appropriate mutation rate. Interestingly, PTSMUT shows equivalent results to PTS, almost no change can be seen by metric $\mathcal{C}$ and $\mathcal{S}$. It seems that the neighborhood of tabu search is able to compensate changes in the neighborhood function to a certain degree, since no deterioration or improvement can be observed. A disadvantage of mutation compared to the tabu search neighborhood function cannot be found as the infeasible mutations do not play an important role.

SPEAN has the same trade-off between the parameters population size, neighborhood size and the number of iterations as the other neighborhood-based methods.

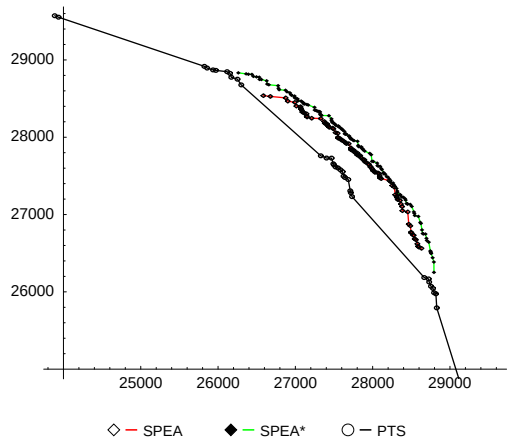


Figure 5.2.: Trade-off fronts for the two objective knapsack problem concerning SPEA with the tabu search neighborhood function.

Because every member of the neighborhood must be evaluated by the objective function $\boldsymbol{f}$, the total number of objective function $\boldsymbol{f}$ evaluations has to be shared among these parameters. Several trade-off experiments by doubling and halving of parameters were performed. The neighborhood size had to remain much smaller than of the one used in PTS, because evolutionary algorithms need a bigger population than local search based methods such as PTS. The elitist archive size was always set equal to the population size. The following characteristic parameter configurations will be presented here:

Configuration	Population size	Neighborhood size	Iterations
SPEA	250	1	2000
1.	250	20	100
2.	60	20	416
3.	60	5	1666

The parameter configurations followed different ideas: Configuration 1 uses the neighborhood in combination with the population size of SPEA, Configuration 2 uses a smaller population, hence having more iterations and Configuration 3 uses a reduced population with a reduced neighborhood, thus having even more iterations. Neither of the parameter configurations could reach trade-off fronts as broad as that of PTS (metric $\mathcal{S}$). An improvement of the convergence compared to SPEA can be seen by metric $\mathcal{C}$ for the configurations 2 and 3.

Above experiments show that the exchange of the neighborhood functions mainly effects the convergence to the Pareto-optimal front and not the broadness of trade-off fronts. The straight forward introduction of the tabu search neighborhood with a size > 1 does not show a positive effect at all, thus it is not clear whether the combination of the concepts of evolutionary algorithms and tabu search can have a favorable effect with this problem.

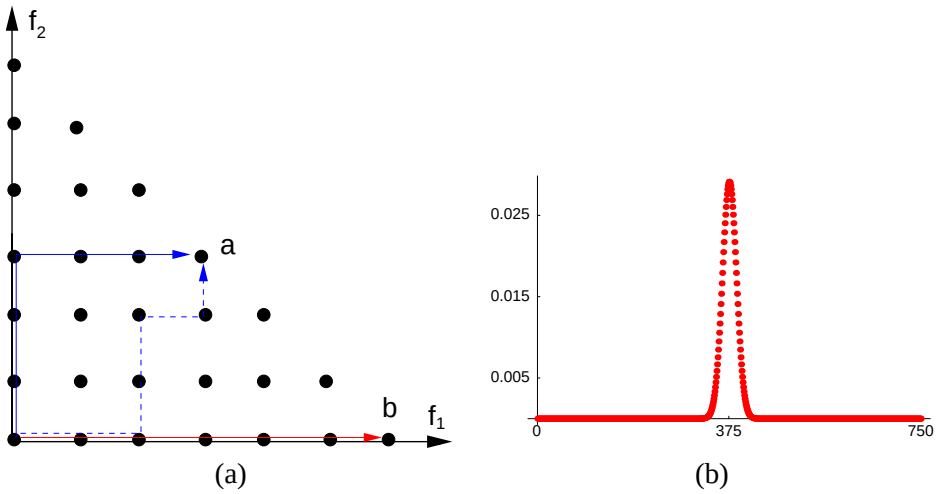


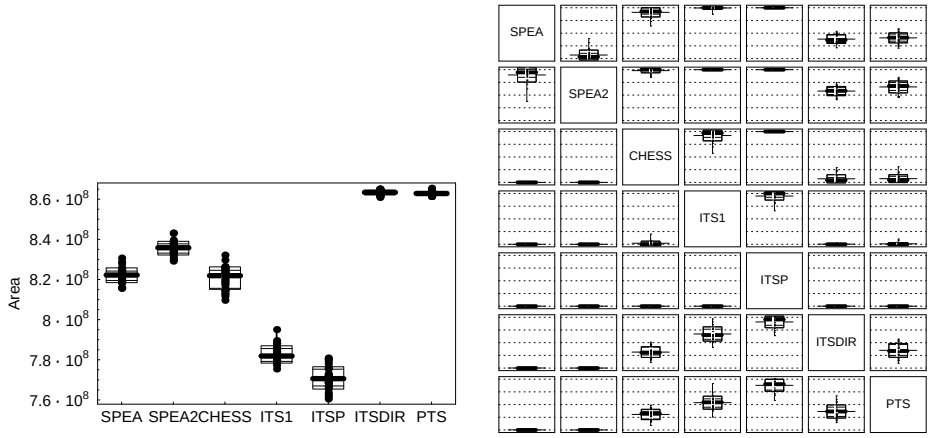
Figure 5.3.: Picture (a) shows the objective space of a simple two objective knapsack problem. Picture (b) shows the binomial distribution for 750 item problem.

5.2. Random Walk Phenomenon

Experiments of section 5.1 could not explain or remedy narrow trade-off fronts of evolutionary algorithms. Interestingly, results of chapter 4 show two groups of methods either generating narrow or broad fronts. Narrow fronts are generated by the Pareto-based methods SPEA, SPEA2 and CHESS. Pareto-based methods control the optimization process upon Pareto-dominance criteria. Broad fronts are obtained by the weight-based methods ITS, ISA, PSA and PTS which are steered by weights. This observation leads to the question: Why are trade-off fronts of Pareto-based methods narrower?

A difference between these two groups is the steering of the optimization process. Directions of optimizations are recognizable by weight-based methods. A run of a weight-based method maintains a specific optimization direction to a certain region of the Pareto-optimal front. On the other hand Pareto-based methods do not seem to maintain optimization directions. New solutions are accepted if they are non-dominated by previous solutions, beside other criteria. Such an acceptance may represent a random optimization direction. Assuming that a search direction can always be determined and that a new accepted non-dominated solution has a random direction in each iteration, then the narrowness of trade-off fronts can be explained by the random walk phenomenon. Solutions perform a random walk in objective space whereas the probability to find solutions at the “ends” of the trade-off fronts decreases.

The random walk phenomenon can be illustrated by a constructed two objective knapsack example. There are two types of items which can be filled into a knapsack of capacity c. Items of type I_1 give a profit 1 in the first objective and 0 in the second. Vice versa profits are obtained by items of type I_2 having profit 0 in the first and 1 in the second objective. Each item has a weight of 1. There are $\frac{c}{2}$ items of type I_1 and $\frac{c}{2}$ items of type I_2. The objective space for a knapsack problem with capacity 6 is



(a) (b)

Figure 5.4.: Distributions of metric $\mathcal{S}$ and $\mathcal{C}$ values concerning the random walk experiments.

depicted on Figure 5.3 (a). A point in the figure represents a knapsack filled with items I_1 and I_2 whereas the number of items I_1 corresponds to the value achieved by f_1 and the number of items I_2 correspond to f_2, respectively. For example point "a" shown in Figure 5.3 (a) contains 3 items I_1 and 3 items I_2 and point "b" contains 6 items I_1. The Pareto-optimal front can easily be found for this simple knapsack problem, i.e., those knapsacks having the sum of the first and the second objective equal to the capacity, $f_1 + f_2 = c$. In order to find the Pareto-optimal front, the following method is used:

1. Start from the empty knapsack.
2. Add randomly one item I_1 or I_2 to the knapsack repeatedly until the capacity is reached.

Step 2 can be seen as a repetition of 1-bit mutations towards the Pareto-optimal front. Figure 5.3 (a) shows that by using this method several search paths to point "a" exist whereas point "b" can only be reached by one path. Since there are several ways to knapsack "a", this point is more probable than point "b" to be found by this method. The number of paths to a point i on the Pareto-optimal front (consecutively numbered from the top) in this constructed example is given by the number of combinations $\binom{c}{i}$. Thus, a probability function for each point on the Pareto-optimal front can be given, i.e. the binomial distribution. The binomial distribution for an example with a knapsack capacity of 750 is depicted in Figure 5.3 (b). This method is only able to find a very narrow front, the most probable points, large parts of the Pareto-optimal front cannot be found since the probability is almost 0. The random walk phenomenon can clearly be seen.

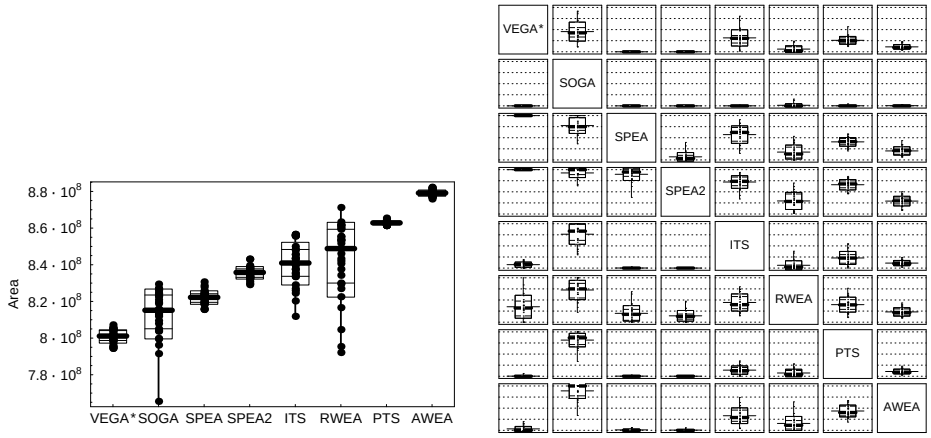
The random walk phenomenon was also experimentally investigated using method ITS on the two objective knapsack problem as presented in Section 4.1. Two experiments without specific optimization directions were performed, in expectation of showing the random walk phenomenon. First, random weight setting of ITS was modified so that a new random weight was set every iteration. The new method is re-

ferred to as ITS1. Secondly, ITS was modified to accept a new current solution upon Pareto-dominance. Instead of selecting the "best" solution of the neighborhood using a weighted sum, a random non-dominated solution is chosen, where this solution must be non-dominated by all other neighboring solutions. This method is called ITSP. Results, depicted in Figure 5.4, fulfill the expectations. Methods ITS1 and ITSP, both without specific optimization directions, show the random walk phenomenon. Visual comparisons of the generated non-dominated fronts showed narrow trade-off fronts. Metric $\mathcal{S}$ reports the smallest values compared to all tested methods so far.

Concluding, it can be said, methods showing the random walk phenomenon are not capable of generating broad trade-off fronts for the knapsack problem. Results of the PTS method show that the random walk phenomenon can be overcome on the multi-objective knapsack problem maintaining specific optimization directions. Optimization directions seem to be important for the multi-objective knapsack problem in order to find large trade-offs. Weights are a simple technique for expressing different optimization directions. Weight-based scalarization of the multiple objectives using different weights proved to be capable of maintaining specific optimization directions on the knapsack problem. But it is questionable, if weight-based methods are capable of finding as good trade-off fronts for problems with more irregular search spaces as for the multi-objective knapsack problem, since the setting of appropriate weights for problems having irregular, complex search spaces can be very difficult.

5.3. Weight-Based Multi-Objective Evolutionary Algorithm

Since modifications of evolutionary algorithms reported in Section 5.1 are not able to overcome the random walk phenomenon, investigations aiming to widen the trade-off front were done. One obvious approach is to introduce weights in evolutionary algorithms in analogy to PTS in order to obtain specific optimization directions. So far, existing weight-based evolutionary algorithms do not maintain specific optimization directions and thus show the random walk phenomenon, e.g. the algorithm of Hajela and Lin (1992). The comparative case study of Zitzler and Thiele (1999) reports narrow trade-off fronts for the algorithm of Hajela and Lin (1992). The method of Hajela and Lin (1992) encodes the weight vectors in the genotype and therefore these are evolved and modified throughout the optimization process. Modifications on existing evolutionary algorithms and several experiments on the two objective knapsack problem were considered in order to investigate the practicability of a weight-based evolutionary algorithm maintaining specific optimization directions. A weight-based evolutionary algorithm (WEA) was developed. WEA is an extension of the elitist VEGA* (vector evaluated genetic algorithm) presented and discussed in Zitzler et al. (2001). The principle of VEGA* was modified in order to enable specific optimization directions. Two variants of a weight-based multi-objective evolutionary algorithm (WEA), denoted as RWEA and AWEA were chosen for comparisons. RWEA uses preset random weights, AWEA uses the adaptive weight setting Algorithm 6 of page 18 in order to determine optimization directions. Parameters for the computational experiments were set corresponding to Chapter 4. The number of specific optimization directions was set to 5 analogous to the 5 runs of PTS. Obtained results of RWEA and



(a) (b)

Figure 5.5.: Distributions of metric $\mathcal{S}$ and $\mathcal{C}$ values resulting from computational experiments concerning the weight-based multi-objective evolutionary algorithms.

AWEA, depicted in Figures 5.5 and 5.6 will be discussed first, the modifications and the experiments which led to these variants will be discussed later.

Comparison

Results show that specific optimization directions can be maintained by evolutionary algorithms using weighted sums, hence the random walk phenomenon does not appear and broad trade-off fronts can be found. Moreover, AWEA achieves the biggest area according to metric $\mathcal{S}$, bigger than PTS, the best method of the comparison presented in Section 4.2. Differences between AWEA and PTS can be found on visual inspections of a trade-off fronts (cf. Figure 5.6). As opposed to PTS the weight-based evolutionary algorithm does not show holes and regions resulting from separate optimization runs. This fact can be attributed to the evolutionary algorithm search principle where no separate runs are performed. The convergence to the Pareto-optimal front of AWEA is mostly better than that of PTS, which is also confirmed by metric $\mathcal{C}$. Almost 50% of PTS are covered by AWEA. Corresponding to section 4.2, adaptive weight setting is able to direct the optimization to the different regions of the Pareto-optimal front, thus AWEA achieves a bigger area than RWEA, seen by metric $\mathcal{S}$.

AWEA compared with the evolutionary algorithms VEGA*, SPEA and SPEA2 finds a significantly larger area (cf. Figure 5.5). But not an as good convergence to the Pareto-optimal front as VEGA*, SPEA and SPEA2 is achieved (metric $\mathcal{C}$), since AWEA does not dominate any solutions found by VEGA*, SPEA and SPEA2, whereas VEGA*, SPEA and SPEA2 cover about 25% of AWEA. Still a better convergence than the PTS method can be stated, AWEA covers more than 30% of PTS; in addition, PTS is covered by SPEA2 with more than 60% compared to the 25% of AWEA.

A comparison of WEA with a weight-based evolutionary reference algorithm also maintaining optimization directions was made in order to verify the performance of

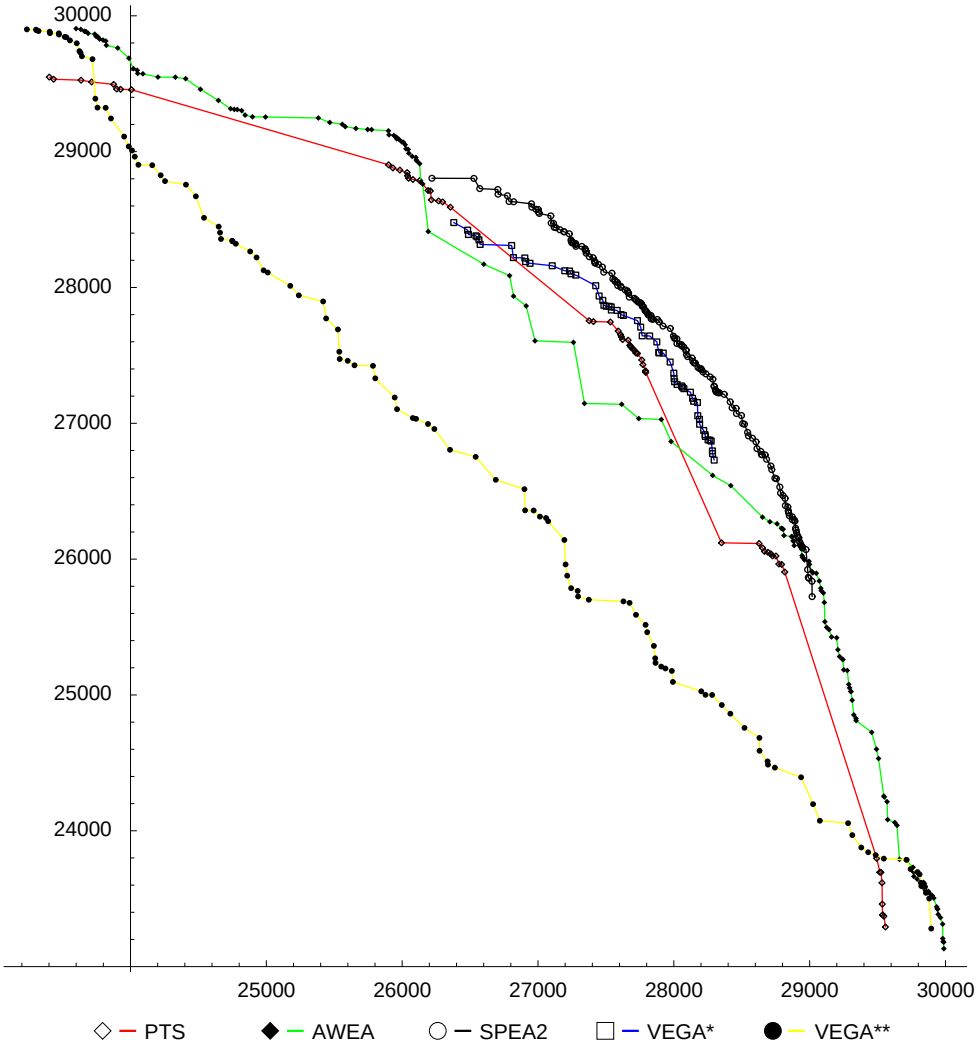


Figure 5.6.: Trade-off fronts for the two objective knapsack problem.

WEA. The reference algorithm called SOGA (single-objective genetic algorithm) is a weight-based evolutionary algorithm, where the population is divided into subpopulations at start-time. The subpopulations are independently optimized towards separate regions of the Pareto-optimal front. For comparative tests, 5 subpopulations having 50 individuals were optimized. SOGA was started with random preset weights. Since SOGA performs independent runs, it can be assigned to the class of independent runs according to the classification of section 3.3. A comparison of SOGA must be made with RWEA, because both methods are started with randomly preset weights. The results, seen in Figure 5.5 show that SOGA is inferior to RWEA. SOGA achieves the smallest area among the methods, hence, no broad trade-off fronts can be generated by SOGA. The worst convergence is also achieved by SOGA, neither of the other fronts can be covered by SOGA, whereas fronts of SOGA are dominated with more then 75%. Thus, performing one run of the multi-objective evolutionary algorithm WEA and sharing information among the individuals is more effective than several independent SOGA runs.

Modifications and Tests

Modifications which were tried out in order to obtain a weight-based evolutionary algorithm are described here. Terms, notations and definitions for evolutionary algorithms are used from (Zitzler et al. 2001). Our weight-based evolutionary algorithm is an extension of the elitist VEGA* (vector evaluated genetic algorithm) presented and discussed in Zitzler et al. (2001). The principle of VEGA* was modified in order to enable specific optimization directions. Instead of switching objectives, true weighted sums are used. The switching objectives of VEGA* represent a special case of weighted sums, where only weight vectors of the identity matrix are used. In a generation, d weighted sums are used. The mating pool is divided into d parts of equal size; part i is filled with individuals that are chosen by tournament selection from the current population according to the weighted sum i. In contrary to VEGA* the mating pool is not shuffled. Crossover and mutation are performed as usual.

In order to obtain a weight-based evolutionary algorithm as good as possible, many computational experiments were performed. First experiments were done with random preset weight vectors using a binary tournament with $d = 5$. Trade-off fronts obtained by this method proved to be even narrower than those of SPEA, obviously no specific optimization directions could be maintained. In order to support directions, experiments increasing the selection pressure were performed. Computational experiments with tournament sizes of 2, 10, 20, 100 and 200 by a given population size of 250 were done. Interestingly, the higher the tournament size the bigger area was achieved (metric $\mathcal{S}$). Therefore a tournament size of 200 was chosen for further experiments. WEA with preset random weights and a tournament size of 200 gives the RWEA method which is discussed above. A high selection pressure has a similar effect to the neighborhood of tabu search. The mating pool is filled after tournament selection with a few solutions appearing many times, these few solutions are mutated to many different offsprings, similar to the current solutions of the different runs of the PTS method. The effect of a high selection pressure can be seen in Figure 5.6 which shows a trade-off front obtained by VEGA**. VEGA** is VEGA* with tournament size of 200 and

without shuffle. A broad front can be seen as opposed to normal VEGA*, but with mostly bad convergence due to only optimizing the objectives.

Adaptive weight setting in analogy to PTS was investigated using Algorithm 6. Algorithm 6 calculates a weight vector $\boldsymbol{\lambda}$ for an individual $\boldsymbol{x}$ regarding the population set X_t. In order to divide the mating pool in d parts, d weight vectors for the weighted sums must be determined by Algorithm 6. Since d is much smaller than the population size, d representatives have to be selected. Three types of representatives $\boldsymbol{x}$ were considered:

- The representatives are chosen randomly from the population.
- The best individuals regarding the weighted sum using the previous weight vectors are taken.
- The representatives are determined by clustering.

New representatives were chosen every iteration in the experiments. The cluster algorithm of SPEA was used for clustering. Additionally, three types of sets X_t for weight setting were considered:

i. The set of representatives themselves,

ii. the whole population, and

iii. the current non-dominated front of the population, whereas the representatives are also chosen from this front.

Computational experiments for these variants, in total 9, were started. Experiments considering clustering had to be dropped because the clustering algorithm of SPEA was computationally to slow. WEA using the best weighted sum individuals as representatives was not able to generate useful trade-off fronts, only the pure objectives f_i were optimized. However, adaptive WEA with random representatives was able to generate broad trade-off fronts. The random representatives with the different sets X_t achieved different results according to metric $\mathcal{S}$, thus showing, that different directions were taken. Areas for the three set X_t types are reported by metric $\mathcal{S}$ in the following order: i, ii and iii, where the first achieved the largest area. Hence, the method AWEA is equivalent to WEA with random representatives and set X_t type i.

The current version of the weight-based multi-objective evolutionary algorithm has conceptual drawbacks:

- The high selection pressure is an abuse of the tournament size parameter introducing a neighborhood similar to that of the tabu search.
- Adaptive weight setting seems not to be optimal. The determination of representatives is only a patch enabling the use of Algorithm 6. Moreover, random determination of representatives for adaptive weight setting is probably not optimal.

Experiments avoiding such a high selection pressure were performed without success. A method with binary tournament was tried out where the part i of the mating pool is

filled with the best individuals according to weighted sum i. A much smaller trade-off front than that of AWEA was obtained.

Because of the convergence problems and the conceptual drawbacks, the methods RWEA and AWEA can only show the practicability of overcoming the random walk phenomenon for multi-objective evolutionary algorithms. Hence RWEA and AWEA cannot be proposed as general methods for multi-objective problems.

6. Conclusions and Outlook

Conclusions

The goal of the present thesis was to find, classify and compare existing multi-objective approaches, which form an alternative to multi-objective evolutionary algorithms. A performance comparison based on the multi-objective 0/1 knapsack problem of alternative and multi-objective evolutionary algorithms was conducted and finally, potential improvements by combinations of optimization concepts were considered. In the following list the key results of this thesis are summarized:

- Eight local search-based methods were found and divided into three classes. All these local search based methods approximate the Pareto-optimal front by performing several runs. The different ways in which each method handles the runs were used as a classification criteria. The following three classes were identified: independent runs, dependent parallel runs and dependent subsequent runs.

- Computational experiments on multi-objective 0/1 knapsack problems showed that the class of dependent parallel runs based on the tabu search, performing adaptive weight setting, yielded the best results among the alternative methods. The comparison between alternative methods and MOEAs showed that weight-based local search methods could find broader trade-off fronts than the multi-objective evolutionary algorithms. The Pareto-based methods under consideration were only able to generate narrow trade-off fronts for the multi-objective 0/1 knapsack problem. MOEAs achieved a better convergence to the Pareto-optimal front on the two objective knapsack problem than the alternative methods.

- The narrow trade-off fronts obtained by Pareto-based methods were ascribed to the random walk phenomenon. The random walk phenomenon describes the case where the trade-off solutions, which are currently improved by the method, do not follow a specific optimization direction, but the solutions pursue a random walk to the Pareto-optimal front. On a simple, constructed knapsack example, it was shown that the probability to find boundary trade-offs was significantly lower than the probability of solutions lying in the center of the trade-off front. On the contrary, the weight-based methods did not show the random walk phenomenon on the knapsack problem, due to using weights which directed the optimization process to certain regions of the Pareto-optimal front. But it is not clear if weight-based methods also perform better than Pareto-based methods on other problems, which have more complex search spaces, since the setting of appropriate weights can be very difficult.

- Three kinds of combinations of methods were implemented and experimentally tested, in order to overcome the random walk phenomenon. First, mutation and the tabu search neighborhood function were exchanged in methods, secondly the tabu search neighborhood was introduced into SPEA and thirdly an experimental weight-based multi-objective evolutionary algorithm was constructed. The first combination did not show any evident improvements or deteriorations of trade-off fronts. The second combination did not yield better results either. Finally, the weight-based multi-objective evolutionary algorithm showed that MOEAs are also able to maintain specific optimization directions, overcoming the random walk phenomenon. The weight-based multi-objective evolutionary algorithm was able to find an as broad front as obtained by the alternative methods. However, the current weight-based multi-objective evolutionary algorithm has convergence problems in the center of the trade-off front.

Concludingly it can be said that the experimental data gave no evidence that the MOEAs could be better than other approaches for multi-objective problems. Nevertheless, multi-objective evolutionary algorithms remain a promising approach for multi-objective optimization problems due to their natural population-based structure.

Outlook

Based on this work, promising topics for future research may be:

- It would be of interest to pursue the random walk phenomenon that was identified during the analysis of Pareto-based methods. Methods avoiding the random walk phenomenon would have to be developed. Since weight-based methods performed well on the multi-objective knapsack problem, the development of a general adaptive weight-based multi-objective evolutionary algorithm maintaining specific optimization directions seems to be a promising approach. Investigations into Pareto-based methods, especially MOEAs, which overcome the random walk phenomenon are needed.

- Since the MOEAs make a better usage of information available during the optimization process such as density consideration of the archive than the alternative methods, the alternative methods seem to have a potential for improvement by using additional information available during the optimization process.

References

Borges, P. C. (2000). CHESS–Changing Horizon Efficient Set Search: A simple principle for multiobjective optimization. *Journal of Heuristics 6*, 405–418.

Czyzak, P. and A. Jaszkiewicz (1997). The multiobjective metaheuristic approach for optimization of complex manufacturing systems. In G. Fandel and T. Gal (Eds.), *Multiple Criteria Decision Making. Proceedings of the XIIth International Conference*, Volume 448 of *Lecture Notes in Economics and Mathematical Systems*, Hagen, Germany, pp. 591–592. Springer.

Czyzak, P. and A. Jaszkiewicz (1998). Pareto Simulated Annealing—A metaheuristic technique for multiple-objective cominatorial optimization. *Journal of Multicriteria Decision Analysis 7*, 34–47.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons.

Ehrgott, M. and X. Gandibleux (2000). An annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum 22*(4), 361–460.

Engrand, P. (1997). A multi-objective approach based on simulated annealing and its application to nuclear fuel management. In *Proceedings of the Fifth International Conference on Nuclear Engineering, Nice, France*, pp. 416–423.

Fonseca, C. M. and P. J. Fleming (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation 3*(1), 1–16.

Gandibleux, X., N. Mezdaoui, and A. Fréville (1997). A tabu search procedure to solve multiobjective combinatorial optimization problems. In R. Caballero, F. Ruiz, and R. Steuer (Eds.), *Proceedings of the Second International Conference on Multi-Objective Programming and Goal Programming*, Volume 455 of *Lecture Notes in Economics and Mathematical Systems*, pp. 291–300. Springer.

Hajela, P. and C.-Y. Lin (1992). Genetic search strategies in multicriterion optimal design. *Structural Optimization 4*, 99–107.

Hansen, M. P. (1997a). Experiments on the usage of hashing vectors in multiobjective tabu search. Paper presented at the NOAS '97, Copenhagen, Denmark, August 15–16 1997. URL: `http://imm.dtu.dk/~mph/papers/`.

Hansen, M. P. (1997b). Solving multiobjective knapsack problem using MOTS. Paper presented at MIC 97, Sophia Antipolis, France, July 21–24 1997. URL: `http://imm.dtu.dk/~mph/papers/`.

Hansen, M. P. (1997c). Tabu search for multiobjective optimization: MOTS. Paper presented at the 13th MCDM Conference, Cape Town, South Africa, January 6–10 1997. URL: `http://imm.dtu.dk/~mph/papers/`.

Hansen, M. P. (1998). *Metaheuristics for multiple objective combinatorial optimization*. Ph. D. thesis, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby Denmark. Report IMM-PHD-1998-45.

Hansen, M. P. (2000). Use of substitute scalarizing functions to guide a local search based heuristic: The case of moTSP. *Journal of Heuristics 6*, 419–431.

Jaszkiewicz, A. (1997). A metaheuristic approach to multiple objective nurse scheduling. *Foundations of Computing and Decision Sciences 22*(3), 169–184.

Sait, S. M. and H. Youssef (1999). *Iterative Computer Algorithms with Applications in Engineering*. IEEE Computer Society Press.

Serafini, P. (1994). Simulated annealing for multi objective optimization problems. In G. H. Tzeng (Ed.), *Proceedings of the Tenth International Conference on MCDM: Expand and Enrich the Domains of Thinking and Application*, Taipei, Taiwan, July 19–24 1992, pp. 283–292. Springer.

Steuer, R. E. (1986). *Multiple Criteria Optimization: Theory, Computation, and Application*. John Wiley & Sons.

Suppapitnarm, A., K. A. Seffen, G. T. Parks, and P. J. Clarkson (2000). A simulated annealing algorithm for multiobjective optimization. *Engineering Optimization 33*(1), 59–85.

Tuyttens, D., J. Teghem, P. Fortemps, and K. V. Nieuwenhuyze (2000). Performance of the MOSA method for the bicriteria assignment problem. *Journal of Heuristics 6*, 295–310.

Ulungu, E. L., J. Teghem, P. H. Fortemps, and D. Tuyttens (1999). MOSA method: A tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis 8*, 221–236.

Viana, A. and J. P. Sousa (2000). Using metaheuristics in multiple-objective resource constraint project scheduling. *European Journal of Operational Research 120*, 359–374.

Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimizations: Methods and Applications*. Ph. D. thesis, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland. Shaker Verlag, Aachen, Germany, ISBN 3-8265-6831-1.

Zitzler, E., K. Deb, and L. Thiele (2001). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation 8*(2), 173–195.

Zitzler, E., M. Laumanns, and L. Thiele (2001). SPEA2: Improving the strength pareto evolutionary algorithm. Technical report, TIK-Report No. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich. May, 2001.

Zitzler, E. and L. Thiele (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation 3*(4), 257–271.