Estimation of distribution algorithms in logistics: Analysis, design, and application

I n a u g u r a l d i s s e r t a t i o n
zur Erlangung des akademischen Grades eines Doktors der
Wirtschaftswissenschaften der Universität Mannheim

vorgelegt
von
Jörn Grahl
aus Dortmund

Dekan:	Prof. Dr. Hans H. Bauer
Erstberichterstatter:	Prof. Dr. Stefan Minner
Zweitberichterstatter:	Prof. Dr. Daniel J. Veit
Tag der mündlichen Prüfung:	12.12.2007

Für meine Eltern
Inge und Achim Grahl

Acknowledgement

This thesis summarizes my research at the Department of Logistics at Mannheim University from 2005 to 2007. Now that I write this introduction, it feels like I started working on my thesis yesterday. Yet, it sometimes felt like a never ending story. This is why I am grateful to many extraordinary people who guided my work into fruitful directions and gave me reliable advice.

First and foremost, I am deeply indepted to Prof. Dr. Stefan Minner for supervising my work. Stefan has taught me *much* more than I expected to learn and maybe more than he expected to teach me. His creativity, dedication and enthusiasm for research are outstanding. I will not forget his openness for new ideas, risky approaches and innovative research fields. It is always of great value to discuss with him. He managed to catch up with my thoughts and problems, provided the solution I was looking for or gave the problem a new spin, no matter how specific or strange the problem was. It is his dedication to rigor and standards as well as his extraordinary efforts in supporting his PhD students that have shaped my view on how research work should be done. I am really looking forward to further working with Stefan and I am happy that I had such a good advisor.

Dr. Peter Bosman's PhD dissertation was among the first material that I read on estimation of distribution algorithms (EDA) years ago. I did not understand a single word. It was great luck for me that Peter answered the many questioning emails with which I bothered him. When I understood more about EDA, we began to work together and successfully finished many research projects. I am delighted that I have the opportunity to work with Peter and hope that we will continue discussing our visions of evolutionary algorithms, music and life in general in the years to come.

Prof. Dr. Franz Rothlauf's support was a steady and reliable one during the last years. He became both a mentor and good friend, and I am glad that he was willing to work and spar with me. Franz had good and right advice when I needed it, not only concerning research-related things. I am deeply indepted to him for the various opportunities he opened up for me and his trust in my capabilities.

Furthermore, I thank Prof. Dr. Daniel Veit for reviewing this thesis, in spite of, and in addition to, his regular workload. Last but not least, Prof. Dr. Armin Heinzl triggered my enthusiasm for research and without this initial trigger, I probably would not have pursued a PhD.

I spent much time with my colleagues at the Department of Logistics. The creative and productive atmosphere makes the department a great place to be for young researchers. I thank Jan Arnold, Stefan Busch, Daniel Dittmar, David Francas, Sebastian Hild, Steffen Klosterhalfen, Mirko Kremer, Lena Kunze, Holger Stephan, Sandra Transchel, Dr. Joachim Vaterrodt and Tobias Weiblen for the many hours of brainstorming, discussion, criticism, work, in short: fun that we had together. Although many different characters and interests meet (did I say collide? ;-)) at the department, we always jointly tried to go "one step further than usual". I loved this enthusiastic spirit and find it unmatched by other institutions until now. I am completely sure that this thesis does not mark an end to our joint projects, neither research work nor going out for beers.

Dr. Joachim Vaterrodt taught me Linear Programming when I was a student years ago. It was great to team with him as a PhD student, to learn from his experience in routing, scheduling, LATEX, and staying cool, and to enjoy his joyful and relaxed character. It was very kind and really helpful, that Gabriele Eberhard and Ruth Pfitzman provided me with alternative literature ("real books!") when I did not want to read one more article!

I dedicate my thesis to my parents. They and my two brothers have blindly supported my academic career until now, never calling me mad when I talked about "estimation of distribution algorithms" and "evolutionary algorithms", topics which are often completely unrelated. They always encouraged me to pursue my goals, stood back often and are there whenever I need them.

Finally, I thank Silvia. During the stressful times of finishing my dissertation you might have gotten the impression that research is the most important and amazing part of my life. It is not. You are.

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

Nowadays, managers are forced to excel competition on a global scale. Efficient business processes have become a basic requirement to meet this goal. However, the design of efficient business processes and the improvement of existing ones is a difficult task that regularly overburdens human intuition. This is especially true in the field of logistics and supply chain management (SCM) where the global integration of economies, the advances in information and communication technologies and increasing customer demands drive the complexity of information, material, and cash flows that require coordination.

In order to streamline and optimize business processes, the scientific field operations research (OR) has developed quantitative approaches since the 1950s. The application of these approaches by industry is nowadays greatly facilitated through the dispersion of advanced planning systems (APS, see Fleischmann and Meyr (2003)). APS integrate data retrieval and management services with quantitative OR planning methods. They can propose optimal or high-quality decisions used in strategic, tactical and operational supply chain planning, e.g., when the optimal strategic network configuration, safety stock allocation, transport plans or lot-sizes are being searched for. As a result, supply chain planners acknowledge the value of quantitative OR planning approaches as essential tools which support decision making processes.

Classical OR approaches encompass, e.g., using the simplex method for solving linear programs or designing branch and bound or dynamic programming algorithms for solving mixed-integer linear programs, see Hillier and Lieberman (2005). Gradient-based root-finding algorithms like the Newton-Raphson method, see (Judd (1998)) solve certain continuous, non-linear optimization problems.

However, the complexity of planning problems that companies face today results in a complicated interplay of conditions under which decision alternatives are optimal. When classical OR approaches are used to solve such problems, in-depth problem-specific knowledge is required in order to avoid in-acceptable runtimes of these algorithms. But, problem-specific knowledge that reduces the combinatorial complexity successfully is often hard to obtain in an appropriate time-span.

Moreover, many real world problems are non-linear, dynamic and uncertain. Economies of scale in production or distribution systems or capacity pooling

effects result in non-linear cost. In order to approximate such problems using (mixed) integer programming solution techniques, non-linear functions require linearization. Linearized problems are hard to solve due to additional integer variabled and resulting combinatorial complexity, in effect the negative impact of linearization on the runtime of classical approaches can easily rule out its use. Classical root-finding algorithms for non-linear optimization can only be applied to a very limited set of problems. They are mislead by local optima, outliers and noise and can not be applied if the required derivatives of the objective function can not be obtained in closed analytical form.

In order to overcome the drawbacks of classical OR methods, OR has broadened its scope significantly and nowadays also embraces a multitude of heuristic approaches for problem solving, see Michalewicz and Fogel (2004) for an introductory review. The goal is to develope efficient methods that are capable of obtaining high-quality solutions in an acceptable timeframe, thereby relaxing the necessity to find an optimal solution but an approximation thereof.

Evolutionary computation (EC, see Eiben and Smith (2007)) is a prominent class of heuristic search methods. EC puts forward algorithms that are inspired by biological principles like survival of the fittest, mutation of genes, recombination and genotype-phenotype mappings. A phenotype is the outward appearance of an actual solution to a problem in the real world. A genotype is a formalization of this solution encoded on by a string of characters (bits, integer values or real numbers) of certain length. The more prominent evolutionary algorithms are genetic algorithms (GA, see Holland (1975) and Goldberg (1989)), evolution strategies (ES, see Rechenberg (1973)) or estimation of distribution algorithms (EDA, Mühlenbein and Paaß (1996)). Evolutionary algorithms (EA) have successfully enlarged the class of planning problems that can nowadays be solved efficiently, see, e.g., Giacobini et al. (2007) and Cotta and van Hemert (2007).

This thesis centers around the analysis, design and application of estimation of distribution algorithms. They are a novel, rapidly developing branch of EC. What differentiates EDA from other meta heuristics is their means to generate candidate solutions to an optimization problem at hand.

Typically, classical and evolutionary optimization algorithms are hand-tailored towards a specific optimization problem. As a result, they resort to a fixed bias towards solutions that exhibit desired properties. This bias is hard-coded into the variation operators used to construct candidate solutions. The intention is that the algorithm is able to solve a specific problem reliably and efficiently. It is obvious that the application of such problem-specific methods to a different planning problem is problematic. For a different problem, the algorithm might generate worse results because its search bias might not fit the structure of the problem. This might result in emphasizing problem features that, although superior in one problem, are inferior in others.

EDA work differently. They have been specifically designed for black box optimization (BBO). In BBO, the inner mechanics, the structure, and the underlying business processes of an optimization problem are hidden from the optimization routine. The only information that can be exploited by the optimizer is a quality measure that is assigned to candidate solutions and a genoytpe-phenotype mapping.

BBO problems arise, if the objective function is not given in closed analytical form. This is frequently the case, if the quality of a solution is estimated from runs of a simulation model. Furthermore, BBO can be a reasonable course of action if some information on the problem is available, e.g., an exact formulation of a convex hull that surrounds all feasible solutions in a mixed-integer linear program. Still, this information might not be sufficient to allow the formulation of a strategy that traverses the search space efficiently and finds the optimum. Additional structural information on the interplay and dependencies between decision variables might be required.

EDA contain built-in learning facilities that intent to extract such structural information from the black box and exploit it during the optimization run. The basic EDA paradigm is discussed in the following. Most evolutionary algorithms involve stochastic elements like randomly generated initial solutions or modification steps that are applied with a certain probability. As a result, the solutions that they generate can be regarded as a random sample path in the search space. This sample follows a probability distribution that is implicitly coded into the variation operators.

EDA make the use of a sampling distribution explicit. They employ a probability distribution which holds for solution components the probability of their appearance in high quality solutions. The distribution is sampled to generate candidate solutions. It is important to note that the distribution is rebuilt iteratively during an EDA run. As the search focuses on high-quality regions of the search space, it propagates elements that are required for solutions to be superior. This is achieved by assigning these components a higher probability. Conversely, low-quality components vanish as they receive a low probability of being sampled. The probability distribution is the major source of variation. It adapts itself towards superior structural elements during optimization. This renders EDA particularly useful for black box optimization.

Up to now, the majority of published material on EDA for discrete optimization is of technical nature, focuses on theoretical aspects and has largely been published in machine-learning and/or EC-theory outlets. Amongst the theoretical concepts that have been developed in this regard are tools that help to analyze the interactions between decision variables in optimization problems. Analysis of problems from logistics by means of these concepts should provide valuable insights into the problems' structure. However, a transfer of these methodological

concepts into the problem domain logistics and SCM has not been pursued, yet.

Furthermore, the application of discrete EDA to problems of logistics and supply chain management is tempting. An application of EDA appears fruitful if the problem considered is non-linear and thus hard to tackle using standard approaches. Furthermore, as EDA and other EA are designed for BBO, it is interesting to study problems where the mainstream solution approaches are hand-tailored towards specific classes or instances of the problem. Efficient BBO algorithms like EDA might substantially reduce the cost of algorithm design and adaptation. Yet, albeit the fact that state-of-the-art EDA are matured problem solvers, applications of EDA in logistics and SCM are rare (if existent at all).

Moreover, the very first EDA have been developed for combinatorial discrete BBO. Motivated by successful results in this domain, the EDA paradigm has been transferred into the continuous domain as well. It was noted though, that a direct adaptation of the EDA principles to continuous spaces does not directly lead to successful results. In fact, initial continuous EDA were unable to solve simple optimization problems that were tractable for hill-climbers.

Note, that in contrast to the discrete domain, most published work on continuous EDA is experimental. It is necessary to derive sound theoretical insights into the optimization behavior of continuous EDA. Such formal results are not available. Although these results are interesting on their own behalf, the major goal must be to exploit the insights and design more efficient continuous EDA. Along this line, it is of special importance to build up an understanding what differentiates discrete and continuous problems from an EDA perspective. Experimental comparisons of the newly designed EDA on commonly accepted test functions and problems that are relevant in practice are needed in order to assess whether the performance indeed increases.

Summarizing, the central research questions that are adressed in this thesis for discrete EDA are:

1. Which insights can be gained from using EDA theory to analyze the structure of certain logistics problems?
2. How efficient and effective are discrete EDA for certain optimization problems in logistics?

The central research questions that are adressed for the continuous problem domain are:

1. How can the optimization behavior of continuous EDA be modeled formally?
2. How efficient and reliable are new, improved continuous EDA which exploit the insights gained from these formal models?

1.2. Goal

The goal of this thesis is twofold. In its first part, EDA theory and EDA themselves are applied to problems from logistics. The goal of this part is to bridge the gap between EDA-theory/application and logistics. The thesis will not design discrete EDA adaptations that are state-of-the-art routines for solving certain benchmark sets. Instead, it will emphasize the ease of EDA applicability across different problems and show that EDA are able to solve complicated problems to global optimality without much adaptation. This illustrates the power of EDA as BBO routines.

The analysis of problem structure will strive to answer the following questions. They will be used as guidelines in the first part of the thesis.

- Is the optimization problem decomposable or separable?
- How do the decision variables interact?
- Which features of phenotypes cause dependencies between in the genotypes?
- Is there a natural way to obtain a genotype-phenotype mapping that encodes dependent bits closely to each other? What is the impact of such an encoding, and what is the drawback if such an encoding can not be designed?
- What is the structure of dependencies in benchmark instances?

EDA theory is used to analyze the structure of warehouse location problems. The warehouse location problem considers the strategic choice of a companies distribution warehouses. A set of customers or market regions and deterministic demand forecasts for a single product are given in conjunction with potential warehouse locations. A set of warehouses must be installed such that the resulting sum of installation costs for warehouses and linear transportation costs between warehouses and customers is minimal. The structure of uncapacitated warehouse location problems is analyzed with the questions above mentioned in mind. It is found, that the numbering of warehouses has a significant impact on the efficiency of simple GA, but not on the efficiency of multivariate EDA. In addition to explaining this numbering defect formally, an experimental study is presented that compares the performance of a state-of-the-art EDA (the Bayesian optimization algorithm, BOA, see Pelikan (2002)) and the sGA on selected instances.

Network design decisions like the warehouse location problem have a long-term impact on a companies business processes as they frame subsequent tactical decisions like safety stock allocation in a supply network. Assume, that some of the installed warehouses face stochastic end-customer demand for a product and a supply network exists that links material, cash and information flows. An important tactical question is where to locate safety stock in the supply network

in order to protect against demand uncertainty. The goal of the resulting safety stock allocation problem is to choose the locations and sizes of safety stock such that a pre-defined service level is reached at minimal cost of holding safety stock. Although state-of-the-art approaches are designed for general networks, the literature in this field is dominated by topology-specific approaches, thereby reflecting the historical development of the field. This thesis takes a different approach and compares the efficiency of evolutionary BBO on safety stock allocation problems in serial, divergent, convergent and general network topologies. The simple GA, the BOA and the (1+1)-EA (a simple stochastic hill-climber) are used in an experimental study that indicates the superiority of the EDA paradigm over the chosen EA in this problem class.

Lot-sizing problems are operational problems that are solved to obtain the cost-optimal sizes of procurement, production or distribution batches while a trade-off between fixed and variable cost components must be balanced. Fundamental single-product dynamic demand lot-sizing problems and dynamic joint replenishment problems will be studied, focusing on the decomposition of these problems. It is shown that they are separable in the meaning of EDA theory. Furthermore, an experimental scalability analysis is conducted in order to approximate the growth of the runtime of EDA with respect to the size of the problem. The results are consistent with EDA scalability theory and indicate the potential of EDA in this field. Mixed-integer linear programming models of these lot-sizing problems can be solved using standard industry solvers even for practically relevant sizes. Thus, the main goal of this study is to obtain structural insights, and not to beat runtime-benchmarks.

In the second part, the thesis centers around the analysis and design of continuous EDA. One goal of this part is to provide formal models that explain convergence of continuous EDA. In a next step, the insights that have been gained from these models will be used in order to develop new and more efficient continuous EDA.

Therefore, the overall convergence process is decomposed into three phases. Theoretical insights are gained from models that concentrate on one of the convergence phases each. The results obtained were previously not available and substantially improve the understanding of what is required for continuous EDA to work well.

Along this line, the adaptation of the EDA principle from the discrete domain to the continuous domain is scrutinized and the major differences between continuous and discrete problem domain for EDA are discussed. It is shown that a direct adaptation of the EDA principle to the continuous domain can easily cause premature convergence on local optima. In this respect, a simple, yet effective, variance scaling policy is integrated into a continuous EDA. The efficiency of the newly designed algorithm is evaluated on standard test functions and in a case study that considers the stochastic transportation problem. It is found, that the implementation of variance scaling enlarges the class of functions that EDA are

able to solve reliably. Additionally, it improves the efficiency of the algorithm on functions that it was already able to solve.

Moreover, the sampling variance trajectory of continuous EDA is analyzed. It is shown that optimal sampling variances exist in simple settings. This result has some impact on future EDA design which is discussed.

Additionally, results on the runtime of a simple continuous EDA on the sphere function are obtained. To be more precise, a closed form expression for the number of generations required to solve the sphere function to a given precision is derived. This result can be a starting point for upcoming theoretical performance comparisons with other optimization algorithms.

1.3. Structure

The following paragraphs contain detailed information on each chapter. The current chapter sets the stage for the content to come by discussing the background, the goal and the structure of this thesis. In Chapter 2, fundamentals are discussed that are required for a thorough understanding of the thesis and a literature review on discrete and continuous EDA is given. Therefore, Section 2.1 introduces basic notation and symbols as well as the simple genetic algorithm (see Goldberg (1989)) and a general framework for estimation of distribution algorithms. It is then discussed in Section 2.2.1 how the notion of problem decompositions and the use of search distributions in EDA are related to each other. Section 2.2 reviews binary estimation of distribution algorithms. It follows the historical development of EDA closely. Along this line, simple EDA from Section 2.2.2 which can not model dependencies between decision variables are followed by EDA that can capture bi-variate interactions in Section 2.2.3. The most advanced EDA are able to model multivariate dependencies between decision variables. They are reviewed briefly in Section 2.2.4. This fundamental chapter is concluded in Section 2.5.

In Chapter 3, the uncapacitated warehouse location problem (WLP) is considered. A subtle, yet important defect is discovered in Section 3.2.2 that can arise when genetic algorithms are used to solve WLP. It is found that the numbering of warehouses has a significant impact on the performance of the GA. This counterintuitive effect is explained in Section 3.2.3 using the factorization theorem that has been developed in Mühlenbein and Mahnig (1999). A large empirical study illustrates in Section 3.2.4 that this malfunction is likely to arise when solving WLP. EDA are proposed as alternative solution methods in Section 3.3. They do not suffer from the numbering defect, and are found to be more efficient on the benchmark instances used.

The placement of safety stock in supply networks is covered in Chapter 4. Safety stock allocation is one field of research where problem specific algorithms for

certain supply network topologies dominate the literature. We challenge the mainstream approach of designing problem-specific algorithms and apply EDA instead. Therefore, a binary encoding is used (see Section 4.3) that exploits an extreme point property. It is shown, that serial safety stock allocation problems are separable in terms of EDA theory in Section 4.4. Experimental results are presented for serial (Section 4.5.1), divergent (Section 4.5.2), convergent (Section 4.5.3) and general acyclic (Section 4.5.4) network topologies. Although EDA regard the safety stock allocation problems as black boxes, they are found to reliably solve the benchmark instances to global optimality. Furthermore, for complicated networks that are particularly relevant from a practical standpoint, EDA clearly outperform the simple genetic algorithms and a stochastic hill-climber in terms of reliability.

Finally, operational lot-sizing problems are studied in Chapter 5. Using the factorization theorem, it is shown in Section 5.3 that single-product dynamic demand lot-sizing problems and dynamic joint replenishment problems are decomposable in the sense of EDA theory. Consequently, a state-of-the-art EDA is used in Section 5.4 to solve both problems in an experimental scalability analysis. The running times measured in fitness evaluations are found to grow with a low-order polynomial that depends on the problem size. This result is in accordance with existing EDA scalability theory.

Chapter 6 lays the foundation for the analysis of continuous EDA. It presents notation and algorithms that are used in the second part of the thesis. Furthermore, it illustrates the decomposition of the convergence process into three phases.

Concentrating on phase 1, Chapter 7 models the population statistics of a simple EDA when it is searching far from the optimal solution. Therefore, it is assumed that the EDA is traversing a monotonous region of the search space that has a slope-like function. Population statistics in a specific generation are derived in Section 7.5, the limit behavior when the number of generations tends to infinity is assessed in Section 7.6. The results show that continuous EDA can easily get stuck in a local optimum, because their sampling variance decreases too fast. The sampling variance directly describes the area of interest that an EDA is currently exploring. A natural way to combat premature convergence is to increase the sampling variance in order to enlarge this area. Chapter 8.1 formalizes the question whether this enlargement can be of arbitrary size, or whether variances exist that should be preferred. It is shown for a simple setting, that indeed sampling variances exist that maximize the proportion of solutions that are optimal. Phase 3 of the decomposition is studied in Section 8.2. In this Section, the number of generations that a simple EDA requires to solve the sphere function to pre-defined precision is derived formally. This result can act as a starting point for a principled comparison of the performance of continuous EDA with other non-linear optimization techniques.

The analytical results require interpretation. Furthermore, they should be exploited in order to design better EDA. This is the major intent of Section 9. Section 9.2 discusses the major differences between discrete and continuous EDA in the light of the formal findings. It is found that indeed a systematic difference between the two exists which triggers different approaches to algorithm design in the discrete and continuous domain. Henceforth, a simple remedy to prevent premature convergence on local optima with virtually no computational overhead is proposed in Section 9.3. It is integrated in an existing continuous EDA, yielding the correlation-triggered adaptive variance scaling IDEA (CT-AVS-IDEA). The basic idea of CT-AVS-IDEA is to enlarge the variances on slope-like regions of the search space. CT-AVS-IDEA is experimentally evaluated in Section 9.4. Its performance is compared to that of the evolution strategy with covariance adaptation (CMA-ES, see Hansen and Ostermeier (2001)), the current state-of-the-art in evolutionary non-linear optimization, and an EDA without variance adaptation. It is found that CT-AVS-IDEA is comparable to CMA-ES and significantly improves upon the state-of-the-art in continuous EDA. CT-AVS-IDEA can solve complicated continuous non-linear optimization problems efficiently and reliably.

In order to demonstrate the applicability of CT-AVS-IDEA beyond artificial benchmarks, it is applied to the stochastic transportation problem (STP) in Section 10. The STP arises in supply chain management, when coordinated decisions on stock levels and shipment quantities must be made under demand uncertainty. It is found that CT-AVS-IDEA routinely generates optimal or near-optimal solutions for the biggest partof benchmark instances used, but shows slight worsening in performance for larger instances investigated.

The thesis is summarized in Section 11.1. Its main achievements are concluded in Section 11.2. Future research possibilities are pointed out in Section 11.3.

2. Fundamentals and literature

In this chapter, we give an introduction to estimation of distribution algorithms (see Mühlenbein and Paaß (1996)). Estimation of distribution algorithms constitute a novel paradigm in evolutionary computation. The EDA principle is still being labeled differently in the literature: estimation of distribution algorithms, probabilistic model building genetic algorithms (PMBGA), iterated density estimation evolutionary algorithms (IDEA) or optimization by building and using probabilistic models (OBUPM). For the sake of brevity we call this class of algorithms EDA.

EDA have emerged in evolutionary computation from research into the dynamics of the simple genetic algorithm (sGA, see Holland (1975) and Goldberg (1989)). It has been found in this research that using standard variation operators, e.g., two-parent recombination or mutation operators, easily leads to an exponential scale-up behavior of the sGA. This means, that the required time measured by the number of fitness evaluations to reliably solve certain optimization problems grows exponentially with the size of the problem. Loosely speaking, the use of fixed variation operators can easily cause a sGA behavior that moves towards enumeration of the search space.

The failure of the sGA is systematic on certain problems. This has triggered research that replaces the traditional variation steps in a GA. Briefly stated, what differentiates EDA from simple GA and other evolutionary and non-evolutionary optimizers is that the main variation in EDA comes from applying statistical learning concepts as follows:

1. The joint probability density of the selected individuals' genotypes is estimated.
2. This density is sampled from to generate new candidate solutions.

As we will illustrate in this chapter, estimating a density from selected solutions' genotypes and subsequently sampling from it to generate new candidate solutions is a tool to make variation more flexible. This is because density estimation can be regarded as a *learning* process. EDA try to learn the structure of problems. They attempt to adapt their search bias to the structure of the problem at hand by applying statistical- and machine-learning techniques on a population of solutions.

The learning capabilities of EDA render them especially suitable for black-box-optimization. In BBO one seeks to find the extrema of a fitness function, without having a formal representation of the latter. One is solely given candidate

solutions and their fitness values. The fitness function itself is unknown, it is encapsulated in a so-called black box. BBO often appears in practice, if little knowledge on the formal structure of the fitness function is available, and solutions are evaluated with a complex virtual or physical simulation model.

EDA have successfully been developed for combinatorial optimization. State-of-the-art EDA systematically outperform simple Genetic Algorithms with fixed variation operators on a broad range of hard problems, such as deceptive problems, MAXSAT, or Ising Spins, see Pelikan and Goldberg (2003) and Pelikan et al. (2006b). Many problems that are intractable for standard GA can reliably be solved to optimality by EDA within a low-order polynomial number of fitness evaluations depending on the problem size.

Regarding this success for the discrete domain, the EDA principle has been adapted for the continuous domain. Continuous EDA intend to solve for solving non-linear optimization problems in continuous spaces that can not be handled by analytical or classical numerical techniques.

This chapter is structured as follows. In Section 2.1, we briefly review the sGA and present a general framework for estimation of distribution algorithms. We focus on discrete EDA for combinatorial optimization in Section 2.2. Fundamental relationships between problem decompositions and search distributions are explained in Subsection 2.2.1. Subsequently, a literature review on discrete EDA for combinatorial optimization is given. Section 2.3 focuses on EDA for non-linear optimization in continuous spaces. A literature review for continuous EDA is presented in Subsection 2.3.1 and 2.3.2. Consequently, we discuss in Subsection 2.3.3 recent continuous EDA that are different to first algorithms. The chapter ends with concluding remarks. A modified version of this chapter has been published as an introductory tutorial chapter in Grahl et al. (2007b).

2.1. Preliminaries and notation

2.1.1. The simple genetic algorithm

The simple genetic algorithm is a cornerstone in GA-theory. It is a stochastic search strategy that maintains a set of solutions $\mathcal{P}$ of size $|\mathcal{P}| = n$, called the population, throughout the search. The population undergoes generational changes. We denote a population in generation t by $\mathcal{P}^t$. A single solution is referred to as an individual. Each individual has an associated fitness value that measures its quality. The goal of the sGA is to find the individual that has the highest quality. An individual consists of a phenotype and a genotype. The phenotype is its physical appearance (the actual solution to the problem at hand) whereas the genotype is the genetic encoding of the individual. The sGA processes genotypes that are binary (bit) strings of a fixed length l. A single bit string is also referred

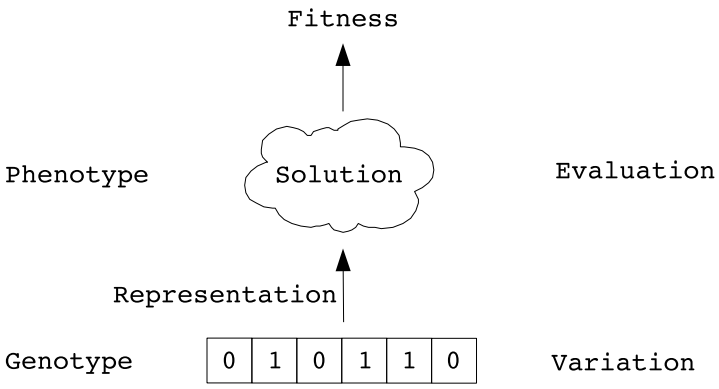


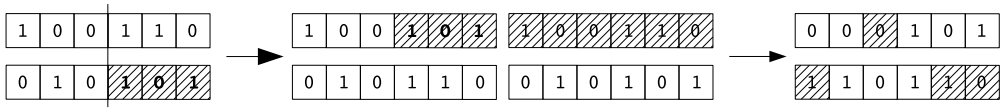
Figure 2.1.: Genotype-phenotype mapping

to as a chromosome. A single bit at position $i, i = 1, 2, \ldots, l$ in the chromosome is also referred to as an allele. The genotype-phenotype mapping is called the representation of the problem and is an important ingredient of GA and EA in general, see Rothlauf (2006). Whereas the fitness of an individual is computed with its phenotype, new solutions are built on the basis of the genotype. For an illustration of the genotype-phenotype mapping, see Figure 2.1.

The sGA processes binary strings of fixed length as follows. The first population of individuals is filled with a random sample from the complete solution space. All solutions are drawn with an equal likelihood, and all genotypes are assumed to be feasible. The fitness of all individuals is evaluated and the better solutions are selected for variation using fitness proportionate selection. In fitness proportionate selection, each individuals probability of being selected is proportional to its quality. Selection intends to push the population into promising parts of the search space. For a formal analysis of selection schemes, see Blickle and Thiele (1996). The set of selected individuals from generation t is called the mating-pool and is denoted by $\mathcal{S}^t$.

New candidate solutions are generated by applying variation operators on elements of $\mathcal{S}^t$. The variation operators of a sGA are recombination operators of the two-parent crossover-type (see Figure 2.2) and bit-flip mutation. For recombination, one-point crossover and uniform crossover are used. New candidate solutions, the so-called offspring $\mathcal{O}^t$, are generated by exchanging partial solutions between the parents. One-point crossover combines parts of two randomly selected individuals (the so-called parents) from $\mathcal{S}^t$ by cutting them into two pieces at a randomly selected locus. Uniform crossover produces offspring by exchanging every single bit between two randomly chosen parents with a predefined probability. Bit-flip mutation modifies single solutions by inverting each bit of the string with a usually small probability.

The offspring $\mathcal{O}^t$ replaces the parents and the next iteration $t + 1$ of the sGA



(a) One-point crossover (b) Uniform crossover

Figure 2.2.: Recombination in the sGA

begins with an evaluation of the newly generated population $\mathcal{P}^{t+1}$. The process of evaluation, selection, variation, and replacement is repeated until a predefined stopping criterion is met, e.g., the optimal solution has been found, the best found solution can not be improved further, or a maximal running time has been reached. See Figure 2.3 for pseudo-code of the sGA.

2.1.2. A general framework for estimation of distribution algorithms

Similar to the sGA, EDA are stochastic, population-based search algorithms. What differentiates EDA from sGA and other evolutionary and non-evolutionary optimizers is that the primary source of variation in EDA is not driven by the application of variation operators to subsets of solutions. Instead, it comes from estimating a probability distribution from all selected individuals $\mathcal{S}^t$ and consequently sampling from this probability distribution to generate offspring $\mathcal{O}^t$. In the following explanation, assume maximization of an objective function.

To illustrate the EDA principle, we introduce a random variable $\mathcal{Z}$ that has an associated probability distribution $P(\mathcal{Z})$ that covers all possible genotypes. To be more concrete, $P(\mathcal{Z})$ denotes a joint probability distribution of all l alleles in the chromosome. A single genotype is denoted with Z, its probability is denoted by $P(\mathsf{Z})$. The random variable associated with a single, $i-$th allele Z_i is denoted by Z_i. The probability distribution of a single allele is denoted by $P(Z_i)$. We further write $P^{\mathfrak{T}}(\mathcal{Z})$ for a probability distribution over all genotypes that has a uniform distribution for all genotypes with an associated fitness value larger

1. Set generation counter $t = 1$
2. Fill $\mathcal{P}^1$ with uniformly distributed solutions
3. Evaluate $\mathcal{P}^t$
4. Select $\mathcal{S}^t$ from $\mathcal{P}^t$
5. Apply variation operators on $\mathcal{S}^t$, fill $\mathcal{O}^t$ with newly generated solutions
6. Replace parents with $\mathcal{O}^t$
7. If termination criterion is met, then **end**,
   else $t = t + 1$, go to step 3.

Figure 2.3.: Pseudo-code for simple genetic algorithm

than $\mathfrak{T}$ and is equal to 0 otherwise. If the probability distribution $P^{\mathfrak{T}^\star}(\mathcal{Z})$ of the optimal solution $\mathfrak{T}^\star$ was known, we would simply have to sample from it to obtain the optimum because all solutions worse than $\mathfrak{T}^\star$ have a chance of zero to be drawn. In practical optimization we do not know $P^{\mathfrak{T}^\star}(\mathcal{Z})$. EDA try to approximate it iteratively.

The first population $\mathcal{P}^1$ of n individuals is usually generated uniformly from all feasible solutions. All individuals are evaluated and the selection step yields a mating-pool $\mathcal{S}^t$ with solutions of higher quality. Now, a probability distribution is estimated from the genotypes of the solutions in $\mathcal{S}^t$. This is achieved by learning a probabilistic model $\mathcal{M} = (\varsigma, \theta)$ from $\mathcal{S}^t$ that is composed of a structure ς and a set of parameters θ. The structure defines (in)dependence between random variables and the associated alleles. Learning the structure ς can be a complex task. The (in)dependence assumptions of the model are chosen such that they match those of the sample $\mathcal{S}^t$ as close as possible. This can be an optimization problem itself. The parameters θ of the model are mostly probabilities and conditional probabilities. They are estimated after the structure that fits best has been found. The model $\mathcal{M}$ represents a probability distribution that approximates the true distribution of the selected solutions' genotypes in $\mathcal{S}^t$. Let $\mathfrak{T}$ denote the worst fitness from the selected individuals. The model $\mathcal{M}$ approximates the true distribution of $P^{\mathfrak{T}}(\mathcal{Z})$, that is the distribution of all individuals that have a better quality than $\mathfrak{T}$. The estimated probability distribution represented by $\mathcal{M}$ is now randomly sampled from to generate offspring $\mathcal{O}^t$. $\mathcal{O}^t$ replaces the worst solutions in the old population $\mathcal{P}^t$, and the population advances to population $\mathcal{P}^{t+1}$. The replacement step uses elitism – it is assured that the best found solution is not replaced by a possibly worse solution. As a result the quality of the best found solution does not decrease over time.

The process of evaluation, selection, model building, model sampling, and replacement is iteratively performed until a predefined convergence criterion is met. Pseudo-code for the general EDA framework can be found in Figure 2.4.

1. Set generation counter $t = 1$
2. Fill $\mathcal{P}^1$ with uniformly distributed solutions
3. Evaluate $\mathcal{P}^t$
4. Select $\mathcal{S}^t$ from $\mathcal{P}^t$
5. Learn probabilistic model $\mathcal{M} = (\varsigma, \theta)$ from $\mathcal{S}^t$
6. Sample offspring $\mathcal{O}^t$ from $\mathcal{M}$
7. Replace the worst solutions in $\mathcal{P}^t$ with $\mathcal{O}^t$
8. If termination criterion is met, then **end**,
   else $t = t + 1$, go to step 3.

Figure 2.4.: Pseudo-code for EDA framework

Note that in other meta-heuristics one is often interested in building efficient

variation operators that are applied on single or subsets of solutions for a specific optimization problem. In EDA the focus shifts from single solutions to the statistical distribution of sets of high-quality solutions in the search space. Loosely speaking, EDA approximate a density function that tells the decision maker where high-quality solutions can be found, which probabilities they have in good populations, and which decision variables are (in)dependent from each other.

In this chapter, we focus on EDA that operate on fixed-length strings because fixed length representation are used in the later chapters of the thesis. A discussion of variable length EDA for Genetic Programming, such as Probabilistic Incremental Program Evolution (PIPE, see Salustowicz and Schmidhuber (1997)), Extended Compact Genetic Programming (eCGP, see Sastry and Goldberg (2003)), or grammar learning approaches, see Bosman and De Jong (2004), are beyond the scope of this chapter.

2.2. Binary estimation of distribution algorithms

2.2.1. Problem decomposition and factorized search distributions

In this section, we focus on the special case that the genotype is a binary string of fixed length l. This means that single alleles have either the value 1 or 0. Although the major results apply to higher alphabets as well, and the proposed algorithms are expandable into this direction (see, e.g., Sastry et al. (2006)), the main stream of research covers the binary case. A simple and straightforward way to implement an EDA that follows the general EDA framework of Section 2.1.2 for binary genotypes of length l would be to use a frequency table of size 2^l as the probabilistic model. The frequency table holds a probability $P(\mathsf{Z})$ for each solution Z. The parameters of this model are the 2^l probabilities of the solutions. These probabilities can be estimated by the relative frequency $\hat{P}(\mathsf{Z})$ of single solutions in the set of selected solutions $\mathcal{S}$ as

$$\hat{P}(\mathsf{Z}) = \frac{\alpha}{|\mathcal{S}|}, \tag{2.1}$$

where α denotes the number of solutions in $\mathcal{S}$ that equal Z. Generating new solutions from this probabilistic model can be done in a straightforward manner by setting the probability to sample Z to $\hat{P}(\mathsf{Z})$ and sample the offspring individual by individual. The structure of this model implicitly assumes that all alleles depend on each other. Using a frequency table of size 2^l exploits no independence assumptions between alleles.

Note that, if we let the population size (and henceforth $|\mathcal{S}|$) tend to infinity, the estimated density expressed by the frequency table converges towards the

true probability distribution $P(\mathcal{Z})$. An iterative procedure of selection, estimation, sampling, and replacement would steadily increase $\mathfrak{T}$ until $P^{\mathfrak{T}}(\mathcal{Z})$ only has positive probability for optimal solutions and has zero probability for all other solutions. However, this approach is generally intractable because the size of the frequency table and thus the effort to estimate its parameters grows exponentially with the size l of the problem. Also, population sizes are finite in practice.

To overcome the drawback of exponentially growing frequency tables, we can allow for the estimation of factorized probability models. Factorizations of joint densities of several random variables are products of marginal densities defined over subsets of the random variables. Factorizations result from a joint density by assuming statistical independence between random variables. The structure of a probabilistic model relates to the (in)dependence relationships between the random variables. The use of factorizations reduces the number of parameters that have to be estimated. Estimating the parameters of factorized probability models is relatively easy, as the parameters can independently be estimated for each factor, see Lauritzen (1996).

A simple example: We assume that all l distributions of the alleles are independent from each other. Then, the joint distribution of the chromosomes can be expressed as a univariate factorization, see Section 2.2.2. The l-dimensional density is decomposed into a product of l one-dimensional densities. The univariate factorization is defined as

$$P(\mathcal{Z}) = \prod_{i=1}^{l} P(Z_i). \tag{2.2}$$

The *structure* ς of this probabilistic model is fixed. The alleles are statistically independent from each other. The parameters θ of this model are the l probabilities of each allele being 0 or 1. Factorizing the joint probability table thus results in a reduction of dimensionality and, henceforth, probability tables that can be estimated more efficiently without a reduction in precision.

Different types of factorizations have been used in EDA, see Sections 2.2.2-2.2.4. Non-surprisingly, depending on the type of factorization that is used, the corresponding EDA exploits different structures of the optimization problem at hand and exhibits a different type of search bias. In general however, the EDA approach of building a model with respect to a certain factorization-type and sampling it to generate offspring is especially suited when it comes to solve *additively decomposable problems*.

According to Mühlenbein and Höns (2005) the fitness function $f(\mathsf{Z})$ of an optimization problem is additively decomposable if it can be formulated as

$$f(\mathsf{Z}) = \sum_{i=1}^{m} f_i(\mathsf{Z}_{s_i}). \tag{2.3}$$

$f(\mathsf{Z})$ is additively defined over m subset of the alleles. The $s_1, s_2, \ldots, s_m$ are index sets, $s_i \subseteq \{1, 2, \ldots, l\}$. The f_i are sub-functions that are only defined on the alleles Z_j with $j \in s_i$. The sub-functions can be non-linear. The Z_{s_i} are subsets of all alleles. These subsets can overlap.

Equation (2.3) exhibits a modular structure. It consists of m components that can, but may not, be coupled. If the s_i are disjoint, $s_i \cap s_j = \emptyset \ \forall \ i \neq j$, the functions do not overlap and the overall problem is called *separable*. Separable problems can be solved by solving the m sub-problems f_i and summing up the results. Depending on the size of the sub-functions, separation reduces the dimensionality of the search space significantly. Assuming that $|s_i| = k \ \forall \ i$ the dimensionality is reduced from l to k and the size of the solution space is reduced from 2^l to $m2^k$. Function (2.3) is called *decomposable* if some sets s_i, s_j exist for which $s_i \cap s_j \neq \emptyset$. In this case, a strict separation of the sub-functions is no longer possible because a single decision variable influences more than one sub-function.

What makes decomposable problems hard to solve? This is a non-trivial question and several answers can be found in the literature. Most obviously the hardness of the sub-functions directly contributes to the overall complexity of the problem. Deb and Goldberg (1993) discuss problems that are called deceptive. They are hard to solve for GA and EDA. Solving deceptive problems is only possible, if a pre-defined number of decision variables is considered simultaneously. Conversely, relying the search on the fitness contribution of any smaller number of variables must fail. Deceptive problems are often assumed as sub-functions for testing purposes. Deceptive functions are typically harder to solve for GA and EDA than non-deceptive functions. Further, sub-problems can contribute to the overall fitness on a similar scale, or the scaling of the sub-functions can differ greatly. In the first case, all sub-functions of equal importance and convergence towards the partial solutions will happen simultaneously. If the sub-functions are exponentially scaled however, the most salient of them will converge first. The other sub-functions may converge later and some instantiations might already be lost at that time. Additively decomposable functions with exponentially scaled sub-functions are harder to solve for GA and EDA – they require a higher population size, see Thierens (1999). Kallel et al. (2001) discuss whether the size $|s_i|$ of the sets influences the hardness of a problem. This can be the case, if for solving $f_i(\mathsf{Z}_{s_i})$ all associated variables must be regarded simultaneously. It may not be the case however, if interactions are not very strong and only some of the dependencies are important. The size of the sets can thus be a source for the hardness of a problem but the degree of connectivity and importance of the dependencies appears to be a more important source for the GA- or EDA-complexity of a function.

High-quality configurations of alleles that belong to the sets s_i are referred to as building blocks (BBs, see Holland (1975), Goldberg (2002)). It is commonly assumed in GA and EDA literature that functions defined on a single BB are not

further decomposable. This means that for solving a sub-problem defined by f_i, all associated alleles Z_{s_i} have to be considered simultaneously. In experiments, this can be achieved by using deceptive functions as sub-problems, see Deb and Goldberg (1993).

The building block structure of an ADF is called a *problem decomposition*. A problem decomposition indicates, which alleles depend on each other and which are independent from each other. Information on the problem decomposition is also referred to as linkage information, see Harik and Goldberg (1997), Harik (1997), Harik and Goldberg (2000) and Goldberg (2002). Tight linkage is a feature of a representation that encodes alleles belonging to the same sets closely to each other. Loose linkage characterizes a solution representation that spreads alleles belonging to the same set widely over the chromosome.

The relationship between a problem decomposition and the factorization of a search distribution is important. Assume a given population $\mathcal{P}$ that contains high-quality solutions for a decomposable problem. The necessity of simultaneous appearance of certain configurations of alleles within a sub-function will cause a statistical dependency between these alleles in $\mathcal{P}$. Alleles from different sub-functions can be set separately from each other to optimize (2.3), and in general they will be statistically independent from each other in $\mathcal{P}$ except for noise and finite population effects. A central element of factorized probability distributions is the possibility to assume independence between random variables. If these assumptions are exactly in accordance with the decomposition of the problem, then the joint probability of dimensionality l is factorized into several marginal densities of possibly smaller dimensionality - each modeling the distribution of alleles in a sub-function. In this case, the factorization is called exact.

Sampling from an exactly factorized distribution is a powerful tool to solve combinatorial optimization problems, see Mühlenbein and Mahnig (1999). However, efficient sampling is not always possible. In the following paragraphs, we refer to work that has been developed in Mühlenbein et al. (1999) and Mühlenbein and Höns (2005)) to illustrate for which decompositions efficient sampling is possible. Assume that a fitness function of type (2.3) is given and one tries to solve the optimization problem $\mathsf{Z}^\star = \arg\max f(\mathsf{Z})$ by sampling solutions Z from a search distribution. A candidate for the search distribution is the Boltzmann distribution which is given as

$$P_\beta(\mathsf{Z}) = \frac{e^{\,\beta f(\mathsf{Z})}}{\sum_y e^{\,\beta f(y)}}, \tag{2.4}$$

see Mandl (1988) where $\beta \geq 0$ and y denotes the set of all solutions. The Boltzmann distribution has the appealing property, that for increasing β it focuses on global optima of $f(\mathsf{Z})$. For $\beta \to \infty$, only global optima have positive probabilities. Unfortunately, sampling from the Boltzmann distribution needs exponential

effort because the denominator is defined over all possible solutions. This is no tractable search strategy.

If the fitness function is additively decomposable, the sampling effort can sometimes be reduced by sampling from a factorization of the Boltzmann distribution. If it can be shown that for a given fitness function $f(\mathsf{Z})$ the Boltzmann distribution can be decomposed into bounded smaller marginal distributions, sampling candidate solutions from it can potentially be a promising search strategy.

To analyze whether this is the case, we define the sets d_i, b_i, and c_i for the index sets s_i for $i = 1, 2, \ldots, m$ as follows:

$$d_i = \bigcup_{j=1}^{i} s_j, \qquad b_i = s_i \setminus d_{i-1}, \qquad c_i = s_i \cap d_{i-1}.$$

If the following Factorization Theorem (Mühlenbein et al. (1999), Mühlenbein and Höns (2005), Mühlenbein and Mahnig (1999)) holds for a given decomposable function, the Boltzmann distribution can be factorized exactly into some marginal distributions.

Factorization Theorem: Let the fitness function $f(\mathsf{Z}) = \sum_{i=1}^{m} f_i(\mathsf{Z}_{s_i})$ be an additive decomposition. If

$$b_i \neq 0 \quad \forall\, i = 1, 2, \ldots, m \tag{2.5}$$

and

$$\forall\, i \ \geq 2\ \exists\, j < i \text{ such that } c_i \subseteq s_j, \tag{2.6}$$

then

$$q_\beta(\mathsf{Z}) = \prod_{i=1}^{m} P_\beta(Z_{b_i} | Z_{c_i}) = P_\beta(\mathsf{Z}). \tag{2.7}$$

This means, that the overall unconditional distribution $q_\beta(\mathsf{Z})$ can be expressed in terms of a conditional distribution $\prod_{i=1}^{m} P_\beta(Z_{b_i}|Z_{c_i})$ which can greatly reduce parameter requirements. Condition (2.6) is called the running intersection property (RIP). If conditions (2.5) and (2.6) hold, then the Boltzmann distribution can be obtained by an exact factorization of marginal distributions. But, it is only reasonable to sample new solutions from (2.7) in order to solve (2.3), if sampling new solutions from (2.7) is computationally easier than solving (2.3) directly. This is not the case if the marginal distributions are of arbitrary dimensionality, because the sampling effort could then grow exponentially with the problem size l. It is indeed the case, if the size of the sets b_i and c_i is bounded by a constant that is independent of the bit-string length l. Then, the factorization is called polynomially bounded.

The effort of sampling a polynomially bounded factorization is much smaller than sampling the unfactorized distribution. Exactly factorizing a search distribution

with respect to a problem decomposition can lead to a significant reduction in dimensionality of the size of the problems that one attempts to solve.

A major result of EDA theory is that if the factorization of the Boltzmann distribution for a combinatorial optimization problem is polynomially bounded, new solutions can efficiently be generated and an EDA can theoretically solve the problem to optimality with a polynomial number of fitness evaluations (Mühlenbein and Mahnig (1999)). This is an important theoretical result that holds for infinite population sizes and if the exact problem decomposition is known. In Sections 2.2.2 - 2.2.4, we will describe how different EDA attempt to transfer this theoretical result into scalable optimizers. A graphical illustration of the EDA principle is available in Figure 2.5.

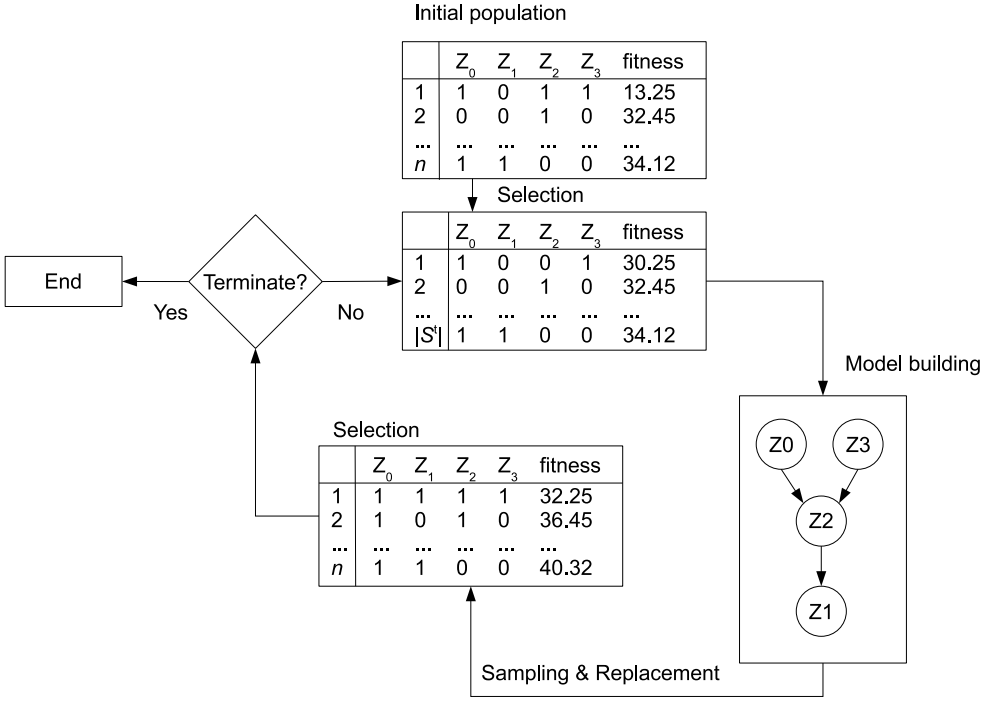


Figure 2.5.: EDA-run as a flowchart. The probabilistic model estimation yields a distribution $P(\mathcal{Z}) = P(Z_0) \cdot P(Z_3) \cdot P(Z_2|Z_0, Z_3) \cdot P(Z_1|Z_2)$.

2.2.2. No interactions

Historically, the first EDA approaches in binary search spaces used univariate factorizations of search distributions. A invariable factorized probability distribution is a product of l univariate probability distributions $\prod_{i=1}^{l} P_{\theta^i}(Z_i)$, where $P_{\theta^i}(Z_i)$ models the distribution of a single allele Z_i and θ^i is the parameter vector that has to be estimated for the i-th allele. θ^i simply holds the probability

of the $i-$th allele being 1 or 0. Note, that univariately factorized probability distributions have a fixed structure. All alleles are assumed to be independent from each other. Univariate factorizations are special cases of general Bayesian factorizations described in Section 2.2.3 with l independent variables.

Estimating a model on the basis of univariate factorizations reduces to parameter estimation because the structure of the density in terms of (in)dependence assumptions is fixed. Estimating parameters for univariate factorizations has a computational complexity of $\mathcal{O}(n|\theta|)$ where n is the population size and $|\theta|$ is the number of parameters that has to be estimated. As the number of parameters that has to be estimated for a single univariate density $P_{\theta^i}(Z_i)$ is usually a small constant, $|\theta| = \mathcal{O}(l)$.

Different EDA based on univariate factorizations of frequency tables can be found in the literature. The bit-based simulated crossover operator by Syswerda (1993) uses a binary string of fixed length l. In the model building process, it computes for each allele i, $i = 1, 2, \ldots, l$ the univariate probability of a 0 and a 1 at bit position i. Therefore, the relative frequencies of single alleles are weighted by taking into account a fitness measure from high quality solutions. New solutions are sampled bit by bit. The sampling process uses the probabilities that were obtained for each bit independently from each other.

Population-Based Incremental Learning (PBIL) by Baluja (1994) uses the same vector of univariate probabilities for each bit like Syswerda (1993). However, instead of re-estimating the complete probability vector from the selected individuals, the probability vector is adapted in each generation from elitist individuals using an update rule. Starting with an initial setting where the probability of each bit being 1 is set to exactly 0.5, the probabilities for each bit are shifted towards 0 or 1. The directions for the shifts are obtained from elitist samples. Offspring is generated bit by bit. The alleles are sampled independently from each other using a univariate probability for each bit.

The compact Genetic Algorithm (cGA) by Harik et al. (1998) is very similar to PBIL. The cGA also works on a probability vector of length l but does not maintain a complete population of solutions. Instead, it updates entries in the probability vectors from two sampled individuals only. The updates are made with an updating rule that is similar to that of PBIL.

In a similar fashion, the Univariate Marginal Distribution Algorithm (UMDA, see Mühlenbein and Paaß (1996)) estimates the probability of being 1 or zero for each bit position from its relative frequency in the population. n new individuals are sampled from the distribution to replace the population as a whole. In contrast to previous work, UMDA does not alter the probabilities after estimating them. New solutions are generated bit by bit from the estimated univariate factorization. The UMDA algorithm approximates the behavior of a simple GA with uniform crossover.

Tsutsui (2002) propose an univariate EDA to solve permutation problems. Binary random variables are introduced for each pairwise combination of permutation elements. The probability associated with each random variable relates to the probability that the associated permutation elements are positioned next to each other in good solutions.

All algorithms discussed in this section are similar to each other and the sGA. They do *not* respect linkage information. This means that sub-solutions can be cut into several parts by the crossover operators and can get lost. The sGA potentially disrupts building blocks in its crossover step. The EDA based on univariate factorization set each bit independently from each other. They do not take into account, that for solving decomposable problems with BBs of size > 1, the joint appearance of configurations of several alleles has to be accounted for.

This is a valid approach, if the problem is separable into sub-problems of order $k = 1$ like the One-Max function. The One-Max function returns the number of ones in a binary string. The simple genetic algorithm with uniform crossover converges on One-Max in $\mathcal{O}(l \log l)$ evaluations, see Mühlenbein and Schlierkamp-Voosen (1993) or Harik (1999). However, if the size k of the sub-problems grows, the performance of the sGA can easily deteriorate. It is well-known that the sGA scales exponentially with the problem size, if the alleles of a composed deceptive trap function are arbitrarily distributed over the chromosome and the order k of the traps is $k > 3$. In this case, the complexity of the simple GA increases from $\mathcal{O}(l \log l)$ to $\mathcal{O}(2^l)$, see Thierens (1995). This clearly demonstrates the boundaries of genetic algorithms with fixed operators. Loosely speaking, the behavior of exponentially scaling GA moves towards that of complete enumeration of the search space. Note that the effect of generating offspring from a univariately factorized probability distribution is similar to using uniform crossover, see Pelikan (2002). Thus, the EDA discussed in this section are expected to scale similarly like the sGA. They are relatively efficient on decomposable problems with sub-problems of smallest sizes and scale badly on problems where the BBs are of higher order.

For real-world optimization, it is not realistic to assume that the BBs of a problem are always of size 1. Linkage information is often not known a priori. This is especially true for black box optimization. Using simple GA with fixed recombination operators or univariate EDA can thus easily result in an exponential scale-up behavior of the algorithm. Note that this can also happen with other fixed recombination operators, and is not related to using one-point crossover or uniform crossover, see Thierens (1999).

2.2.3. Bivariate interactions

Exponential scalability of univariate EDA on problems with BBs of higher order has motivated the development of EDA based on more involved probabilistic

models. This subsection reviews EDA that are capable of capturing bivariate dependencies between alleles. In contrast to EDA presented in Section 2.2.2, the structure of the probabilistic model of the EDA in this section is no longer fixed. Instead, the structure of the model is flexible to a certain degree, allowing for an adjustment of the factorization that is built in every generation. Flexible probabilistic models allow for an adaptive bias of the algorithm. To be more specific: EDA discussed in this section attempt to convert linkage information into statistical (in)dependence relationships correctly. Learning probabilistic models as done in many EDA corresponds to minimizing the discrepancy between the distribution expressed by the model and the empirical distribution of the alleles in the set of selected individuals. Inside the boundaries imposed by a model type, EDA search for that model that fits the distribution of the genotypes best.

In the binary problem domain, Bayesian factorizations of search distributions based on frequency counts are commonly used as probabilistic models. A Bayesian factorization of a joint density is a product of conditional densities $P(Z_i|Z_{\pi_i})$ for each random variable Z_i. π_i denotes a vector of so-called parents of Z_i that indicates on which variables the univariate density Z_i is conditioned. A Bayesian factorization of a joint density $P(\mathcal{Z})$ is defined as

$$P(\mathcal{Z}) = \prod_{i=1}^{l} P_{\theta^i}(Z_i|Z_{\pi_i}), \tag{2.8}$$

where θ^i denotes the vector of parameters that has to be estimated for each density $P_{\theta^i}(Z_i|Z_{\pi_i})$. The matrix $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_l)$ represents the structure of a Bayesian factorization. $\boldsymbol{\pi}$ has a representation as a directed acyclic graph with l nodes where each node corresponds to a single Z_i, $i = 1, 2, \ldots, l$ and an arc from node j to node i indicates that Z_i is conditioned on Z_j. Importantly, the Bayesian factorization is only a valid density if and only if the factorization graph is acyclic. These probabilistic models are also referred to as graphical models or Bayesian networks, see Heckerman et al. (1994), Lauritzen (1996) and Jordan (1999) for details.

EDA that are capable of capturing bivariate dependencies between variables restrict the general form of a Bayesian network given in (2.8) such that each variable can at most depend on a single other variable. This means, that $|\pi_i| = 1 \; \forall \; i = 1, 2, \ldots, l$.

Estimating the parameters of a given Bayesian factorization is straightforward. The probabilities of bit i of being 0 or 1 is estimated by the relative frequency of this bit of being 0 or 1 given all possible configurations of the parents π_i. This means, that the probability of this bit of being 1 or 0 is held in a frequency table of size $2^{|\pi_i|}$. The frequency table lists all possible configurations of the values of the $|\pi_i|$ parent variables and for each configuration the corresponding relative

frequency of the i-th bit being 1 or 0 in the mating-pool.

Sampling from Bayesian factorizations is done in two consecutive steps. First, the nodes are sorted in topological order. If nodes are visited in topological order, a node i is visited after all of its parent nodes π_i have been visited already. In bivariate EDA, each node has at most a single parent. After the nodes have been sorted, sampling starts at a root node and then visits nodes in topological order. The probabilities of a bit i of being 0 or 1 is set to the corresponding relative frequency that has been calculated in the parameter estimation process.

The Mutual Information Maximization Input Clustering algorithm (MIMIC, see De Bonet et al. (1997)) uses a factorization graph that has the structure of a chain. In a chain, each random variable has exactly one parent and conditions exactly one random variable (except for starting and ending nodes in the chain). For selecting the chain that models the distribution of the selected individuals as good as possible, MIMIC minimizes the difference between the distribution expressed by the factorization and the joint distribution of all alleles in the population. Therefore, the Kullback-Leibler divergence between both distributions is taken as a distance measure that is minimized with a greedy chain constructing approach. The Kullback-Leibler divergence is a measure of difference between two probability densities, see Kullback and Leibler (1951). The computational complexity of the model building process is $\mathcal{O}(l|\theta|n)$. As $|\theta| = \mathcal{O}(l)$ in a chain, the overall model building complexity in MIMIC is quadratic in l.

The Combining Optimizers with Mutual Information Trees algorithm (COMIT, see Baluja and Davies (1997)) uses a factorization graph that has the structure of a tree. If the factorization structure is a tree, then every random variable $\mathcal{Z}_i$ is conditioned on exactly one parent. In contrast to the chain model of MIMIC, several variables can be conditioned on the same variable. To find a dependency tree that models the distribution of the genotypes of the selected individuals as good as possible, the COMIT algorithm uses a learning technique from Chow and Liu (1968) that results in a maximum-likelihood factorization with tree structure. The computational complexity of the model building process of COMIT is similar to that of MIMIC.

The Bivariate Marginal Distribution Algorithm (BMDA, see Pelikan and Mühlenbein (1999)) uses a set of independent trees as its factorization graph. In this model, each random variable has at most a single parent. The model building process starts with an empty factorization graph. The factorization is constructed greedily by adding edges on the basis of a χ^2 dependency test between pairs of random variables, where the edges are chosen in descending order of the related χ^2 statistic. The computational complexity of the model building process is $\mathcal{O}(l^2|\theta|n)$. As $|\theta| = \mathcal{O}(l)$, the model building complexity is cubic in l.

In contrast to EDA that use univariate factorizations of the search distribution, EDA that allow bivariate dependencies between alleles are more powerful.

However, this comes at the price of a more complex model-building process. Moreover, EDA that can model bivariate dependencies between alleles are still not able to solve general $k-$decomposable problems efficiently. As reported in Bosman (2003), the number of fitness evaluations that an EDA based on tree-like factorization of the joint density requires to successfully solve this problem grows exponentially with the size of the problem l. This is caused by inexact factorizations with respect to the problem decomposition that allows mixing solutions of different sub-problems instead of sampling partial solutions to the sub-problems independently from each other.

2.2.4. Multivariate interactions

Bivariate probabilistic models are not sufficient to solve problems with high-order interactions like deceptive problems. Thus, probabilistic models that are able to capture multivariate interactions between alleles were proposed in order to advance the model flexibility further. Therefore, marginal product factorizations and Bayesian networks without restriction on the network structure were used.

A marginal-product factorization is a decomposition of a joint density into a product of multiple multivariate densities, where the multivariate densities are defined over mutually exclusive subsets of all considered random variables. This means that each random variable appears in a single factor of the product only. We call the set of variables that forms the $i-$th multivariate density a node-vector v_i. The node partition vector $\mathbf{v} = (\mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_m)$ represents the structure of the marginal product factorization. A marginal product factorization is defined as

$$P(\mathcal{Z}) = \prod_{i=1}^{m} P_{\theta^{\mathsf{v}_i}}(Z_{\mathsf{v}_i}). \tag{2.9}$$

Marginal product factorizations decompose a multidimensional density into a product of possibly multidimensional densities. In contrast to Bayesian Factorizations, marginal product factorizations do *not* use conditional densities. They simply assume that all variables that appear in a factor v_i are jointly distributed. Thus, the size of the frequency tables associated with a node vector v_i grows exponentially with $|\mathsf{v}_i|$, whereas in Bayesian factorizations, it grows exponentially with the number of parent variables $|\pi_i|$. Since a single allele can not appear in several factors in the marginal product factorization, this model is suited to modeling problem decompositions with non-overlapping building blocks, that is separable ADFs.

Bayesian factorizations are a more general class of distributions that comprehend that of marginal product models. In Bayesian factorizations, single alleles can be, but do not necessarily have to be, associated with a single marginal density.

Thus, Bayesian factorizations are suitable for problem decomposition in which building blocks are non-overlapping *and* in which they are overlapping.

Marginal product factorizations where the multivariate densities $P_{\theta^{\mathsf{v}_i}}(Z_{\mathsf{v}_i})$ are represented by frequency tables were proposed in the Extended Compact Genetic Algorithm (ECGA, see Harik (1999)). The ECGA builds a marginal product model, starting with a univariate factorization of the joint density of the binary genotype. A greedy heuristic is used to iteratively join marginal densities such that the improvement in a minimum description length (MDL) metric is maximized. MDL metrics are used in order to evaluate the goodness of fit of an explanatory model to statistical data, see Grünwald (2005). The greedy heuristic stops, when no potential merging of marginal densities improves the MDL metric further. The total number of fitness evaluations required by the ECGA to solve additively decomposable problems of order k grows with $\mathcal{O}(2^k m^{1.5} \log m)$ where m is the number of BBs in the problem. Note that the ECGA is able to properly decompose a search density based on frequency tables with respect to a problem decomposition. The time required to solve additively decomposable problems measured by the number of fitness evaluations grows sub-quadratic with the size of the problem if the size of the problem k is bounded independent from l.

To solve permutation problems, marginal product factorizations were used by Bosman and Thierens (2001b) and Bosman and Thierens (2001c). The factorizations were built on permutation random variables that allow for a direct representation of permutations in combination with the greedy model building process of the ECGA. Both the Akaike Information Criterion (AIC) metric and the Bayesian Information Criterion (BIC) metric are used. In contrast to the MDL metric, both the AIC and the BIC metric require parameter tuning in order to prevent over-fitting of data. They differ in how model complexity is penalized, see Burnham and Anderson (2002) for details. The results indicate a sub-quadratic growth of the minimally required population size with respect to the problem size on decomposable deceptive permutation problems. Simple GA scale exponentially on these problems with respect to the problem size.

Acyclic Bayesian networks without further restrictions on the structure of the graph were independently proposed for use in the Bayesian Optimization Algorithm (BOA, see Pelikan et al. (1999), Pelikan et al. (2000), Pelikan (2002)), the Estimation of Bayesian Network Algorithm (EBNA, see Etxeberria and Larrañaga (1999)) and the Learning Factorized Distribution Algorithm (LFDA, see Mühlenbein and Mahnig (1999)).

In these algorithms, the model building starts with a univariate factorization. In univariate factorizations, all variables are assumed to be independent from each other. Iteratively arcs are added to the probabilistic model in a greedy fashion. This relates to assuming dependence between alleles. A first version of the BOA uses a Bayesian-Dirichlet metric (see Heckerman and Geiger (1995)) to

measure the fit between the distribution expressed by the model and the empirical distribution of the alleles in the mating-pool. Greedily, arcs that maximize the Bayesian-Dirichlet metric are added to the graph, where arcs that cause cycles are skipped. If the maximum number of parent variables π_i is bounded from above by κ, then this greedy approach to model building has a computational complexity of $\mathcal{O}(\kappa l^3 + \kappa l^2 |\theta| n)$.

Later versions of the BOA (see Pelikan et al. (2001)) and the EBNA and LFDA use penalization metrics similar to the Bayesian Information Criterion (BIC) metric. Using a greedy way to estimate a Bayesian network from selected individuals with the BIC metric has led to sub-quadratic scale-up behavior of these algorithms on additively decomposable problems, see Mühlenbein and Mahnig (1999) and Pelikan (2002). Scalability results for the BOA that also suits the LFDA and EBNA can be found in Pelikan (2002) and Pelikan et al. (2003). According to BOA scalability theory, the number of fitness evaluations that the BOA requires to reliably solve additively decomposable problems grows between $\mathcal{O}(l^{1.55})$ for uniformly scaled sub-problems and $\mathcal{O}(l^2)$ for exponentially scaled sub-problems independent of k.

Unconstrained Bayesian factorizations with local structures represented by decision graphs were used by Pelikan and Goldberg (2001) in the Hierarchical BOA (hBOA) to solve hierarchically decomposable problems. Hierarchically decomposable problems are decomposable and introduce additional dependencies on several levels of interpretation. The large number of dependencies in these problems can hardly be expressed in a straightforward manner different from decision graphs. In hBOA, the decision graphs are combined with a niching scheme. Niching localizes competition between solutions and ensures that only similar solutions compete with each other. hBOA uses restricted tournament replacement. For each newly generated solution, a set of currently available solutions is picked randomly. The most similar solution is replaced by the new solution, if the fitness of the new solution is higher. The combination of Bayesian factorization with local structures and restricted tournament replacement has lead to sub-quadratic scale-up behavior on hierarchically decomposable problems, see Pelikan et al. (2006a). hBOA was successfully applied to Ising spin-glass problems and MAXSAT, see Pelikan and Goldberg (2003).

The BOA and the hBOA have both been used to solve multi-objective problems, see Khan et al. (2002) and Pelikan et al. (2005). The BOA was able to solve deceptive multi-objective problems that other evolutionary algorithms could not solve. In a practical application, no significant superiority to GA that use classical crossover operators could be found, see Laumanns and Ocenasek (2002). The hBOA is found to scale up well on multi-objective decomposable deceptive problems. To obtain good scale-up however, clustering techniques in the objective space are needed. On the considered test problems, multi-objective variants of the simple GA and the UMDA scale badly and are incapable of solving problem

instances of medium sizes.

As we have seen in this chapter, the use of flexible probabilistic models allows for a flexible bias of the EDA. State-of-the-art EDA that are built on multivariate probabilistic models solve decomposable problems efficiently. Moreover, the number of evaluations required to solve such problem reliably should grow sub-quadratically, given the minimal population size is known. The increase of the model flexibility has lead to more involved structure learning processes. At the same time, the class of problems that can reliably be solved was enlarged such that the resulting EDA consistently outperform standard genetic algorithms with fixed operators on hard decomposable optimization problems.

2.2.5. Summary

Discrete EDA use probabilistic models to guide their search for high quality solutions. State-of-the-art EDA incorporate statistical learning techniques for building a probabilistic model during the optimization and thereby are able to adapt their bias to the structure of the problem at hand. For the important class of additively decomposable problems, the use of Bayesian factorizations or multivariate factorizations has led to scalable optimizers that reliably solve additively decomposable problems within a sub-quadratic number of fitness evaluations.

2.3. Continuous estimation of distribution algorithms

The considerable success of discrete EDA has motivated researchers to adapt the general EDA principle for the continuous problem domain. Continuous EDA are proposed for function optimization in continuous spaces. Their application can be promising if classical numerical methods like gradient-based methods fail or are not applicable because derivatives are not available or due to outliers or noise. Continuous EDA are used for what is often referred to as global optimization. For continuous optimization with the EDA of this section a genotype is a vector of size l of real values if not stated otherwise.

Continuous EDA mostly use variants of the normal probability density function (pdf) as the basis of their probabilistic model because the normal pdf is a commonly–used and computationally tractable approach to represent probability distributions in continuous spaces. The normal pdf $P^{\mathcal{N}}_{(\boldsymbol{\mu},\boldsymbol{\Sigma})}$ for $l-$dimensional random variables $\mathcal{Z}$ is parametrized by a vector $\boldsymbol{\mu}' = (\mu_1, \mu_2, \ldots, \mu_l)$ of means and a symmetric covariance matrix $\boldsymbol{\Sigma}$ and is defined by

$$P^{\mathcal{N}}_{(\boldsymbol{\mu},\boldsymbol{\Sigma})}(\mathcal{Z} = \mathsf{Z}) = \frac{(2\pi)^{-\frac{l}{2}}}{(\det \boldsymbol{\Sigma})^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathsf{Z}-\boldsymbol{\mu})'(\boldsymbol{\Sigma})^{-1}(\mathsf{Z}-\boldsymbol{\mu})}. \tag{2.10}$$

The number of parameters to be estimated from data to fit the normal distribution to selected individuals equals $\frac{1}{2}l^2 + \frac{3}{2}l$. A maximum likelihood estimation for the normal pdf is obtained from a vector $\boldsymbol{\mathcal{S}}$ of samples if the parameters are estimated by the sample average and the sample covariance matrix, see Anderson (2003) and Tatsuoka (1971):

$$\hat{\boldsymbol{\mu}} = \frac{1}{|\boldsymbol{\mathcal{S}}|}\sum_{j=0}^{|\boldsymbol{\mathcal{S}}|-1} \boldsymbol{\mathcal{S}}_j, \quad \hat{\Sigma} = \frac{1}{|\boldsymbol{\mathcal{S}}|}\sum_{j=0}^{|\boldsymbol{\mathcal{S}}|-1} (\boldsymbol{\mathcal{S}}_j - \hat{\boldsymbol{\mu}})(\boldsymbol{\mathcal{S}}_j - \hat{\boldsymbol{\mu}})^T \tag{2.11}$$

On the basis of the normal pdf, different probabilistic models can be estimated from the selected individuals. The resulting EDA are discussed in the following sections.

2.3.1. No interactions

A univariate factorization of a multidimensional normal pdf is a product of several independent univariate normal pdfs. This means that all covariances between two normally distributed random variables $Z_i, Z_j, i \neq j$ are 0. The variance of the $i-$th allele random variable is denote by σ_i^2. Univariate factorizations of normal distributions are defined as

$$P(\mathcal{Z}) = \prod_{i=1}^{l} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-0.5(\frac{\mathsf{Z}_i - \mu_i}{\sigma_i})^2}. \tag{2.12}$$

Rudlof and Köppen (1996) proposed the first continuous EDA based on the normal pdf. Their algorithm is an adaptation of the binary PBIL algorithm (Baluja (1994)) to continuous domains. The distribution of each allele is modeled by a univariate normal distribution. In the initial phase of the optimization run, the variances are set to high values to stimulate exploration of the search space. They are adapted in further generations with a geometrically decaying schedule to enforce convergence. The means are adjusted with a learning rule similar to the original discrete PBIL algorithm. New solutions are generated allele by allele by sampling the values for each allele from the univariate Gaussian distributions independently from each other.

A second adaptation of the binary PBIL algorithm is presented in Servet et al. (1997). In this algorithm, a lower and an upper bound for each variable is given that defines a range where values for the variables lie. A simple histogram model with two bins is maintained. The first bin corresponds to the lower half of the value range, the second bin corresponds to the upper half of the value range. The binary random variables of the PBIL algorithm correspond to the probability that a solution lies in the upper half of the range. For example $P(\mathsf{Z}_4 == 1)$

denotes the probability that the fourth allele has a value in the upper half of the initialization range. If the values converge towards a single bin, then the bin sizes are adaptively re-sized to the half of that range, similar to bisection. Sampling is again done allele by allele.

Sebag and Ducoulombier (1998) propose another adaptation of PBIL to continuous spaces. They use a univariate normal pdf for modeling the distribution of each allele independently from each other as in Rudlof and Köppen (1996). In contrast to the latter, the variance and the mean are updated with the same learning rule.

In Gallagher et al. (1999), the parameters of the univariate factorization are estimated from a set of selected individuals with the standard maximum-likelihood estimator for mean and variance. The approach of Gallagher et al. (1999) is more similar to the EDA principle compared to previous approaches.

Similarly, the Univariate Marginal Distribution Algorithm in the Continuous Domain (UMDA_c) was proposed in Larrañaga et al. (2000a). UMDA_c is an adaptation of the UMDA algorithm by Mühlenbein and Paaß (1996) to continuous domains. In UMDA_c, a first population of candidate solutions is sampled uniformly from the set of feasible solutions. The fitness of each individual is evaluated using f. Now we use truncation selection of the best $\tau \cdot 100\%$ individuals. This means, given a population size of n individuals, we select the $\tau \cdot n$ best individuals. Selection pushes the population towards promising regions of the search space. From these selected individuals the following probabilistic model is estimated. In UMDA_c it is assumed, that the joint distribution of the selected individuals follows a l-dimensional normal distribution that factorizes over l univariate normals. That is, the covariances between all x_i and x_j are 0 for all $i \neq j$. Thus, in generation t, the n variables $X_{1\ldots n}$ follow a univariate normal distribution with mean μ_i^t and standard deviation σ_i^t.

The parameters μ_i^t and σ_i^t are estimated from the selected individuals by using the well known maximum likelihood estimators for moments of the univariate normal distribution.

Now, new individuals are generated by sampling from the estimated joint density. These individuals completely replace the old population. The algorithm continues with truncation selection again. This process is iterated until a termination criterion is met.

2.3.2. Multivariate interactions

To increase the modeling capabilities of the probabilistic model, research into more involved probabilistic models on the basis of the normal pdf was conducted. To be more specific, similar to the discrete domain, the model structure is no

longer assumed to be fixed, but is allowed to be flexible to a certain degree. This is achieved by allowing for modeling multivariate interactions between continuous alleles. Consequently, the probabilistic models comprehend multivariate densities. For real-valued optimization, marginal product factorization, Bayesian factorizations, and mixture-based factorizations have been proposed.

Marginal product factorizations of the normal pdf are products of possibly multivariate normal densities where each allele random variable belongs to exactly one factor. Similar to multivariate factorizations in the discrete problem domain, they are defined for the continuous domain as:

$$P(\mathcal{Z}) = \prod_{i=1}^{m} P^{\mathcal{N}}_{(\boldsymbol{\mu}^{\mathsf{v}_i}, \boldsymbol{\Sigma}^{\mathsf{v}_i})}, \tag{2.13}$$

where $\boldsymbol{\mu}^{\mathsf{v}_i}$ denotes the $|\mathsf{v}_i|$-dimensional mean vector and $\boldsymbol{\Sigma}^{\mathsf{v}_i}$ the covariance matrix of the $i-$th partition. Multivariate factorizations of normals have first been proposed in Bosman and Thierens (2001b).

Bayesian factorizations based on the normal pdf are referred to as Gaussian networks (see Heckerman and Geiger (1995)). For real-valued optimization, the use of Gaussian networks has independently been proposed by Bosman and Thierens (2000) in the iterative density estimation evolutionary algorithm (IDEA)-framework and in Larrañaga et al. (2000a). The latter uses a variant of the MIMIC algorithm based on the normal pdf, called MIMIC_c, and unrestricted Gaussian networks in the Estimation of Gaussian Network Algorithm (EGNA).

As a first approach to learning the Gaussian network Bosman and Thierens (2000) used an algorithm by Edmonds (1976) to build a factorization of the search distribution with minimum entropy. In this factorization, each variable is allowed to depend on at most another variable. In Bosman and Thierens (2001a), unrestricted Gaussian networks are used. A greedy factorization learning scheme in combination with a Bayesian Information Criterion metric is used to learn the structure of the Gaussian network. In Larrañaga et al. (2000a), the model building process starts with a complete factorization graph. Arcs are greedily removed from the factorization based on a likelihood-ratio test.

Mixture distributions are weighted sums of $M > 1$ pdfs. The probabilistic model defined by a mixture distribution is a collection $\boldsymbol{\varsigma}$ of M (simpler) probabilistic model structures ς_m and a collection $\boldsymbol{\theta}$ of M parameter vectors where $m = 1, 2, \ldots, M$. A mixture distribution is then defined as

$$P_{(\boldsymbol{\varsigma},\boldsymbol{\theta})}(\mathcal{Z}) = \sum_{m=1}^{M} \beta_m P_{(\boldsymbol{\varsigma}_m, \boldsymbol{\theta}_m)}, \text{where} \tag{2.14}$$

$$\beta_m \geq 0, \quad \text{and} \quad \sum_{m=1}^{M} \beta_m = 1. \tag{2.15}$$

The factors of this product are called mixture components, the weights β_m are called mixing coefficients. The interesting feature of mixture-based probabilistic models is that they allow to model the distribution of solutions independently on different peaks, potentially allowing a population to concentrate on more than a single peak in the search space. This is of special importance, if the search function is multi-modal and several basins of attraction should be investigated simultaneously. To achieve this, mixture-based probabilistic models for continuous optimization have been proposed by Bosman and Thierens (2001a). Clustering techniques like k-means clustering are used to divide the population into M sub-populations from which the parameters for each of the mixing components are estimated. Maximum likelihood estimates for (2.14) can not be obtained analytically. Instead, an iterative procedure defined in Dempster et al. (1977) is used to obtain the estimates for the mixing coefficients, the mean vector and the covariance matrix.

Real-valued mixture probability distributions have been used for multi-objective optimization in Bosman and Thierens (2002) and Bosman and Thierens (2003). The results indicate that a mixture-based model can effectively help to maintain diversity along the pareto-frontier of a multi-objective problem. Similar observations were made in Costa and Minisci (2003), where pdfs based on Parzen-windows, that are similar to mixture distributions, are used.

Continuous EDA can readily be used to solve permutation problems if a real-valued representation of permutations is chosen, see Bosman and Thierens (2001b), Larrañaga et al. (2002), and Robles et al. (2001). These approaches are however not very effective, as the commonly used Random-Key representation is highly redundant, see Bosman and Thierens (2001b).

2.3.3. Recent approaches

For discrete search spaces, the use of Bayesian factorization based on frequency counts has led to scalable evolutionary optimizers that outperform classical genetic algorithms on a broad range of problems. It has been noted however, that simply estimating and sampling probabilistic models on the basis of the normal pdf does not automatically lead to efficient EDA for the continuous problem domain.

The main problem is premature convergence on local optima when using maximum likelihood-based normal EDA, see Chapter 7, Grahl et al. (2005) and Bosman and Grahl (2008). In order to solve the problem of premature convergence, a class of more involved probability distributions could theoretically

be introduced for use as a search distribution in continuous EDA. However, contours of continuous fitness landscapes can be of virtually any shape. As universal approximation in arbitrary exactness is computationally intractable, recently developed EDA still stick to the use of the normal pdf, but emphasize a more sensible tuning of the parameters of the normal distribution.

Ocenasek et al. (2004) use a self-adaptation approach adopted from evolution strategies to scale the variance after the distribution estimation. The results indicate that the performance of the resulting algorithm is comparable to that of the evolution strategy with covariance adaptation (CMA-ES, see Hansen and Ostermeier (2001)), a state-of-the-art evolution strategy, on separable univariate functions.

Gallagher and Frean (2005) propose a scheme that tries to approximate the Boltzmann distribution (see Section 2.2.1) in continuous spaces using the normal pdf. Note that the Factorization Theorem and the related theoretical work is valid for discrete and continuous domains. Its implications are nonetheless limited to the discrete domain, because a parametric distribution such as the normal pdf might mislead the optimization in the continuous domain, see Chapter 9 for a discussion. The results from Gallagher and Frean (2005) indicate that indeed a more sensible tuning of the parameters of the normal pdf that is not limited to using maximum-likelihood estimates results in more efficient continuous EDA.

Yuan and Gallagher (2005) modify the estimation scheme such that it maintains diversity in the population by restricting the variances to values greater than 1. This reduces the risk of premature convergence, because it directly enlarges the area that the algorithm is exploring.

Grahl et al. (2006) propose to adaptively scale the covariance matrix after the estimation process. The scaling of the variance is triggered on slope-like regions of the search space and is disabled when the currently investigated region is shaped like a peak. To identify which structure currently dominates the investigated region, the use of ranked correlation estimates between density of the normal pdf and the fitness values is proposed. The results on non-linear problems that can not be solved by simple hill-climbing algorithms (e.g., Rosenbrocks function in high dimensions) show that the proposed algorithm scales with a low order polynomial depending on the problem size. Computational results are very close to that of the CMA-ES (Hansen and Ostermeier (2001)), one of the leading algorithms in continuous evolutionary optimization.

Bosman et al. (2007a) extend the work Grahl et al. (2006) by proposing a variance scaling trigger that is especially suitable for high-dimensional problems. In Bosman et al. (2007b), an extension is proposed that tunes not only the variances of the Gaussian model, but also shifts the mean of the distribution.

The above approaches are specially interesting in this thesis, as they largely relay on the normal distribution. However, similar results were found by other

researchers, using different probability distributions than the normal distribution, but still attempting to obtain a maximum–likelihood estimate, see Larrañaga et al. (2000b), Cho and Zhang (2001), Shin et al. (2001), Shin and Zhang (2001), Cho and Zhang (2002), Ocenasek and Schwarz (2002), Paul and Iba (2003a), Paul and Iba (2003b), Ahn et al. (2004), Kern et al. (2004) and Cho and Zhang (2004).

All of the above approaches advance the class of problems that continuous EDA are able to solve reliably. A more sensible adaptation of the EDA principle to the continuous domain seems to be required to develop efficient continuous EDA. Recently obtained results are promising and indicate the potential of continuous EDA when it comes to solve continuous optimization problems. The adaptation of the normal density in the continuous domain appears to be a central element of study for EDA to come.

2.3.4. Summary

The success of discrete EDA has motivated researchers to adapt the EDA principle to the continuous domain. A direct adaptation of the principle has however proven not to be effective. In contrast to the discrete field, recent work shows that the model structure is not as important as a right treatment of the model parameters. A more sensible adjustment or alteration of the density estimation or sampling process for continuous EDA is required to advance the performance of continuous EDA. Recent approaches have proven to be successful and reliable for continuous non-linear optimization problems.

2.4. Ant colony optimization and EDA

Estimation of distribution algorithms belong to the larger class of meta-heuristics. While this class of strategies comprises many different approaches towards heuristic optimization, some of them are closely related to EDA. Following Quinlan (1993) and Zlochin et al. (2004) heuristic algorithms can be classified as being either “instance-based” or “model-based”. The larger part of search algorithms can be classified as being instance-based because they generate candidate solutions on the basis of current solutions or a population of solution. Genetic algorithms and local search techniques with its variants are prominent members of this class. On the other hand, model-based search (MBS) algorithms have been proposed that generate candidate solutions on the basis of a probabilistic model. The probabilistic model captures information that is regarded as crucial in order to guide the search into high-quality regions of the search space. Estimation of distribution algorithms are clearly a member of the latter class. Furthermore, ant-colony systems (ACO, see Dorigo (1992), Dorigo et al. (1997) and Dorigo

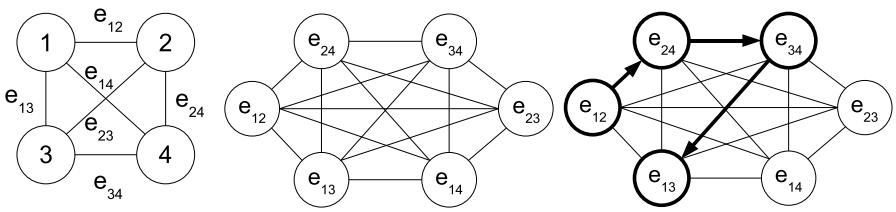
and Di Caro (1999)), share conceptual similarities with EDA that will briefly be covered in the following.

ACO was inspired by the foraging behavior of real ants, see Deneubourg et al. (1990). It was found, that ants are capable of finding near-shortest paths between their nests and food. In the beginning of the search for food, ants explore the surroundings of their nests in a random manner. Once food is found, it is brought back to the nest. During their return, ants place a pheromone trail on the ground. The intensity of this trail can depend on the quality and amount of food that was found. The pheromone trail is used as a guidance by other ants when searching for food. Over time, pheromone trails develop between the nests and places where food can be found. This basic concept is exploited in ACO in order to solve combinatorial optimization problems.

In ACO, solutions are constructed on the basis of a pheromone model. The pheromone model directly relates to a probabilistic distribution over the solution space. Constructed solutions are used such that the pheromone model biases the search towards high quality regions of the search space. The major difference between ACO and EDA is the use of a constructive heuristic in ACO. The constructive heuristic is a problem-specific technique that builds complete solutions sequentially. Starting with an empty solution, parts are added based on a construction rule. In contrast to using the construction heuristics alone, ACO randomizes the sequence in which the construction steps are carried out. The construction process can be modeled as traversing a construction graph. The construction graph consists of nodes that correspond to states. A state defines the current parts of a solutions that have already been built. Arcs that contain construction steps link nodes. The choice of the construction step is dependent on the pheromone value that is associated with this step and usually proportional to this value. This means, that the transition probabilities between states are dependent on the probabilistic model expressed by pheromone values.

The pheromone model is used as an adaptive bias towards construction sequences that result in good solutions. In order to be able to adapt the sequence dependant on the problem at hand, a pheromone update is done. Pheromone updates are basic ingredients of every ACO algorithm, but can vary greatly between implementations. Usually, a pheromone evaporation uniformly reduces the amount of pheromone. This allows to "forget" old solutions and explore larger regions of the search space, thereby adhering to a diversification of the search. In a second step, one or more solutions that have been constructed are used in order to increase pheromone trail parameters that are part of the solutions considered. This update rule can take many forms, see Dorigo and Stützle (2004) for a detailed analysis. The unifying idea of update rules is to increase the probability of using construction steps that are contained in high quality solutions via positive feedback, directly adhering to an intensification of the search.

A simple example taken from Blum (2004) for a 4-city TSP is illustrated in Figure 2.6. Subfigure 2.6(a) contains a graph $G = (V, E)$ of the TSP instance. The edges e_{ij} are solution components that in combination make up a tour. They have distances d_{ij}. Assume that the tour should start in city 1. A first construction step can allow a random edge $e_{1j}, j = 2 \ldots 6$ to be added. In following steps, it must be assured that a Hamiltonian cycle results from the chosen edges. Subfigure 2.6(b) shows the construction graph. Subfigure 2.6(c) shows a single path through the graph that relates to a single solution. In order to use an ACO to solve this problem, the choice of edges could be based on a simple nearest-neighbor selection. It can be randomized by allowing the choice of further-away cities and biased over time towards edges that are available in high-quality tours.



(a) TSP instance (b) Solution graph (c) Construction of solution $< e_{12}, e_{24}, e_{34}, e_{13} >$

Figure 2.6.: Example for ACO TSP solution construction

The main differences between ACO and EDA follow. EDA work on partial solutions directly. They store (conditional) probabilities that are associated with the presence of partial solutions. ACO work on construction steps that add parts to partial solutions. They store probabilities that are associated with the use of construction steps. As a direct consequence, EDA depend on a representation that maps the partial solutions to phenotypes and work on the alleles on a genotypic level. Thus, proper representations on whose genotypes a probabilistic model can be defined are essential for EDA. ACO work directly in phenotypic space. Thus, construction heuristics are essential in order to apply ACO. Furthermore, while multivariate EDA build a probabilistic model that can capture dependencies between decision variables, pheromone trail parameters in ACO are handled as independent probabilities. Conditional probabilities are not (yet) used in ACO. Blum and Dorigo (2005) show that this can systematically lead to premature convergence. Finally, while the use of problem-specific routines allows the design of highly efficient ACO for specific problems (see Dorigo and Stützle (2004)), ACO is not usable for BBO due to the same reason. This is in direct contrast to EDA that are designed with BBO in mind.

2.5. Conclusions and outlook

This chapter intended to serve as an introduction to estimation of distribution algorithms. We discussed the major terms, concepts, and algorithms for the discrete and the continuous problem domain and pointed out the difference between the two fields. State-of-the-art EDA systematically outperform standard GA with fixed recombination operators on hard additively decomposable problems. To be more specific, the total number of fitness evaluations that is required to reliably solve additively decomposable problems to optimality is often found to grow sub-quadratically with respect to the problem size.

In order to choose an EDA for a given discrete optimization problem, the most helpful information is knowledge about the complexity of interactions between decision variables. If most variables are independent, a simple EDA such as the UMDA can be chosen. If no such information is available, it is a reasonable strategy to use a multivariate EDA such as the BOA in a first step and analyze the probabilistic models that it constructs during optimization in a second step. If these prove to be simple, switching to a bivariate or univariate EDA might make sense. If the models feature highly interdependent decision variables, one should keep using a multivariate EDA because simpler EDA are not able to capture these dependencies.

First stage EDA for continuous optimization rely solely on variance estimation and do not tune the variance after estimation. These first stage EDA should be neglected in favor of more recent implementations like the CT-AVS-IDEA, that is presented in Chapter 9.

The correct choice of parameters like the population size, selection intensity or termination conditions can be facilitate using parameter-less implementations of evolutionary algorithms. Parameter-less implementations are available for the simple GA in Harik and Lobo (1999), the ECGA in Lima and Lobo (2004) and for the hBOA in Pelikan and T. (2004). The current state-of-the-art in parameter setting in EA is presented in Lobo et al. (2007).

Most of the publications on EDA come from the computer-science, EC-theory and machine-learning community. This research has lead to effective optimizers and theoretical insights. In industry however, the sGA and its variants are still predominantly used although EDA research has clearly shown us the boundaries of these approaches. This is partly due to the fact that the EDA principle is relatively new and still unknown in industry. In addition to existing applications for EDA, (see e.g. Blanco et al. (2001), Sierra et al. (2001), Larrañaga and Lozano (2001), Bengoetxea et al. (2002), Ducheyne et al. (2002), Pelikan and Goldberg (2003), Blanco et al. (2003), Pelikan et al. (2006b)), more applications of EDA to problems of industrial relevance are needed that illustrate practitioners the effectiveness of EDA and the drawback of classical approaches. Furthermore,

EDA theory on problem decompositions might provide insightful results when being applied to problems of practical interest.

Henceforth, the application of EDA to problems in supply chain management and logistics, as well as the analysis of such problems by means of EDA theory will be at the center of Chapters 3-5.

In the continuous domain, the use of probabilistic models has not directly lead to effective optimizers. In contrast to the discrete problem domain, a more sensible adjustment of the estimated parameters is necessary to avoid premature convergence and boost performance. Most of the available results for the continuous domain are still experimental. Developing formal models that help us to understand the dynamics of continuous EDA is a formidable and promising task. According to this need, theory will be developed, its insights exploited and the outcome experimentally validated in Chapters 6-10.

Part I.

Applications of discrete EDA and EDA-theory in logistics and supply chain management

3. Decomposition of warehouse location problems and the linkage problem

3.1. Introduction

Location decisions have a long-term impact on a company's business processes. By defining production-, warehouse- and distribution sites, they frame subsequent tactical decisions such as safety stock or capacity allocation and operational decisions such as transportation or production planning, just to name a few. It is therefore not surprising that ample research has been conducted to solve location problems of various kinds, for reviews see, e.g., Daskin (1995), Drezner and Hamacher (2002), and Klose and Drexl (2005).

At their core, discrete warehouse location problems are solved to balance cost trade-offs that arise from warehouse opening decisions and resulting material flows and transportation between installed locations and customers. The uncapacitated warehouse location problem (uWLP) is the fundamental discrete WLP and is the topic of this chapter. It determines the optimal number of warehouses from a finite set of potential locations and balances the cost trade-off between opening and resulting transportation cost. uWLPs are $\mathcal{NP}$-hard, see Cornuéjols et al. (1990). Overviews are available in Cornuéjols et al. (1990) and Klose and Drexl (2005). As solution procedures to uWLPs, a variety of exact algorithms have been proposed in the literature. Krarup and Pruzan (1983) attribute the best performance to dynamic programming, cutting plane, pseudo-boolean programming, and combined approaches. Despite efficient algorithms such as DUALOC (see Erlenkotter (1978)), large instances of the WLP and its variants such as WLPs with limited warehouse storage capacity and WLPs that integrate routing decisions for goods delivery can still not be solved in reasonable time. This justifies heuristic methods such as constructive ADD heuristics (Kuehn and Hamburger (1963)), Lagrangian heuristics (Guignard (1988), Beasley (1993)) or meta-heuristics such as simulated annealing and tabu search (see Klose (2001), Hoefer (2002) or Michel and van Hentenryck (2004)). Several genetic algorithms have been proposed to solve the uWLP successfully. Kratica et al. (1996) hybridize a simple GA with an ADD-heuristic. Horng et al. (1999) integrate clustering and local search into a simple GA. Filipovic et al. (2000) study the effect of different selection schemes in simple GA for uWLPS. Kratica et al. (2001) develop a more efficient implementation of the sGA for uWLPs. Jaramillo et al. (2002) provide a comparative study of a GA with elitism, binary tournament

selection and fitness-based fusion crossover used to solve the uWLP and several other location problems.

The existing body of literature is largely experimental, and work is rare that analyzes on a theoretical level the prerequisites under which GA work well/badly on uWLPs. The contribution of this chapter is to link structural properties of the uWLP to results from GA theory. This allows us to draw conclusions on the expected behavior of simple genetic algorithms for uWLPs and present a subtle defect that may arise when using sGA to solve uWLPs. To be more precise, we focus on inherent functional drawbacks of sGA when it comes to learning and exploiting dependencies between the warehouse opening decisions. We label this drawback the "numbering defect". It is relevant for the solution of uWLPs but to the best of our knowledge has not received attention in the literature until now. Furthermore, with estimation of distribution algorithms we propose an alternative evolutionary approach that does not suffer from the defect and outperforms sGA on uWLPs reliably. The insights gained from this chapter are valuable for optimization practitioners who design evolutionary algorithms for location problems.

The remainder of this chapter is structured as follows. The uncapacitated warehouse location problem is described in Section 3.2.1. Sections 3.2.2 - 3.2.4 present experimental results that illustrate defects of simple GA on the uWLP, explain the defects theoretically and, by means of a large numerical study, underline the necessity to overcome them. Section 3.3 proposes EDA as alternative evolutionary algorithms that do not suffer from the defects and solve uWLPs faster and more reliable than sGA. The chapter ends with concluding remarks and an outlook on further research. This chapter was published in Grahl et al. (2007c).

3.2. Linkage in warehouse location problems

3.2.1. The uncapacitated warehouse location problem

The uWLP considers the selection of a non-empty subset of warehouses out of a given set of M potential warehouses such that the sum of transportation costs between warehouses and N customers and opening costs is minimal. Let f_i denote costs of opening warehouse $i = 1, \ldots, M$. A binary indicator variable y_i is set to 1 if warehouse i is built and to 0 otherwise. Let c_{ij} denote transportation costs for delivering the total demand of customer $j = 1, \ldots, N$ from warehouse i. x_{ij} is the fraction of demand that is being shipped from warehouse i to customer j. All x_{ij} follow the single assignment property (see Krarup and Pruzan (1983)), which states that an optimal assignment of customers to a given set of open uncapacitated warehouses is obtained by assigning every customer j to warehouse i for which $c_{ij} = \min\{c_{ij} \mid i = 1, \ldots, M;\ y_i = 1\}$. In presence of uncapacitated

warehouses and linear transportation costs, each customer is assigned to the closest open warehouse which, in turn, supplies him/her completely. As a result, fractions x_{ij} are $\in \{0, 1\}$. A binary programming formulation of the uWLP that follows Bilde and Krarup (1977) is given as follows:

$$\min \quad C = \sum_{i=1}^{M} f_i y_i + \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij} x_{ij} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{i=1}^{M} x_{ij} = 1 \qquad \forall j = 1, \ldots, N \tag{3.2}$$

$$y_i - x_{ij} \geq 0 \qquad \forall i = 1, \ldots, M; j = 1, \ldots N \tag{3.3}$$

$$y_i \in \{0, 1\} \qquad \forall i = 1, \ldots, M$$

$$x_{ij} \geq 0 \qquad \forall i = 1, \ldots, M; j = 1, \ldots, N$$

(3.1) minimizes the total costs of warehouse opening decisions and transportation. Equality constraints (3.2) assure that total customer demand is delivered and (in optimal solutions) assigns each customer to a single warehouse. Opening and assignment decisions are linked in (3.3).

3.2.2. Numbering defects

We used a sGA to solve uWLPS. The sGA employed one-point crossover, bit flipping mutation and fitness proportionate selection. The crossover probability has been set to 1.0, the mutation probability to 0.01. A sGA run was terminated by allele convergence, i.e., for each allele the same value was present for at least 95% of the individuals. We mapped the warehouse opening decisions $y_i, i = 1, \ldots, M$ onto genotypes of fixed length $l = M$. The i-th bit denotes the opening decision of warehouse i. This coding is commonly used by GA that can be found in the literature. Infeasible solutions with no open warehouse are discarded using a large penalty value. We exploit the single assignment property by assigning each customer to his/her closest open warehouse. This representation is illustrated in Figure 3.1. Circles represent customers, solid lined triangles represent open warehouses (warehouses 1,4, and 5) and dashed lined triangles closed warehouses (warehouses 2, and 3). Directed arcs from alleles to warehouses indicate the assignments of the opening decisions to the warehouses.

We consider some uWLP benchmark instances from the well-known ORLIB and Galvao-Raggi test-beds[1] and encode the y_i on the bitstring in the exact order that

[1] All instances that are used in this chapter are available in the uncapacitated facility location library (UFL library), `URL:http://www.mpi-sb.mpg.de/units/ag1/projects/benchmarks/UflLib/`

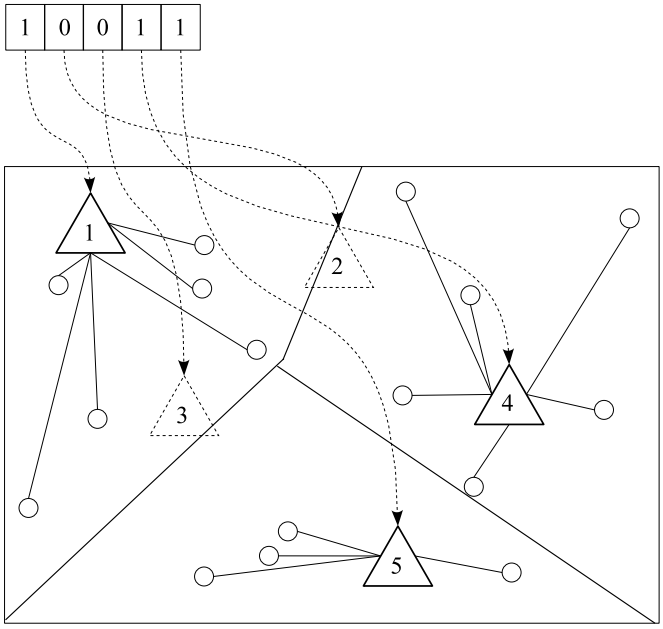


Figure 3.1.: Representation for the uWLP

is proposed in the benchmarks. We determine the smallest population size for which the sGA finds the optimal solution in 27 of 30 independent consecutive runs using bisection. The number of fitness evaluations is averaged over the successful trials, yielding a hardware-independent measure of performance $\bar{e}$. The results are listed in Table 3.1. Instances are named as in the description files.

Test-bed	**Instance name**	M	$\bar{e}$	$\bar{e}'$	**Factor**	$\bar{e}''$	**Factor**
ORLIB	cap71	16	885	906	1.02	795	0.89
	cap102	25	3,595	5,778	1.06	2,165	0.60
	cap132	50	25,636	27,975	1.09	11,731	0.46
Galvao-Raggi	50.1	50	10,710	18,331	1.71	10,248	0.96
	70.1	70	19,117	55,637	2.91	15,899	0.83
	100.1	100	54,112	159,223	2.94	32,586	0.60

Table 3.1.: Number of fitness evaluations with different numbering

We now illustrate what we label the numbering effect. A slight modification is made to the representation. An opening decisions y_i is no longer encoded by bit i of the genotype, but by a different bit. For example, warehouse number 4 could be re-labeled to warehouse number 17, and its opening decision is no longer encoded by bit 4 but by bit 17. Still, every opening decision is encoded by a single bit and each bit encodes a single opening decision. The numbering has been changed systematically in order to obtain the following results. Detailed explanations will be given in Section 3.2.3. Note that, all warehouse-specific parameters and thus the instances themselves (including optimal solutions) do not change. We repeat

the experiment. Results are listed in Table 3.1. The average number of fitness evaluations is $\bar{e}\,'$.

Interestingly, the performance of the sGA deteriorates for all instances. The effect gets worse for the larger instances, where computation times increase by a factor of up to ≈ 3. We perform another renumeration and repeat the experiment. The average number of fitness evaluations is $\bar{e}\,''$, see Table 3.1. This numbering of the warehouses proves to be advantageous. The performance of the GA increases throughout all test instances: up to half the computation times could be saved compared to the initial experiment by simply using a different numbering of the warehouses! Why is this the case?

In the remainder of this chapter, we

1. *explain* this behavior theoretically,
2. show its *existence* empirically, and
3. propose EDA as *alternative* evolutionary algorithms that do not suffer from this defect and are more efficient.

3.2.3. Decomposition of the uWLP

We will use results that link a decomposition of the uWLP to the performance of evolutionary algorithms in order to explain the numbering effect that was presented in section 3.2.2. Therefore, we review some results on problem decompositions and its effect on the scalability of EA and thereafter propose a decomposition of the uWLP in this Section. Empirical results from widely used benchmarks highlight the likeliness of witnessing the outcome of the defect in Section 3.2.4.

Recall, that crossover operators used in a GA that do not disrupt BBs are called linkage-friendly. The use of linkage-friendly crossover operators significantly enhances GA performance (Thierens and Goldberg (1993)). However, to design linkage-friendly crossover operators, the decomposition of the problem has to be known a priori.

An alternative approach to reducing disruption of BBs stems from the order in which bits are encoded on the binary string, see Goldberg et al. (1989). Codings that place bits from a BB closely together on the genotype are said to possess tight linkage and (due to less BB disruption) increase the performance of sGA. In contrast, codings that place bits of a BB widely spread over the genotype are said to possess loose linkage. They can trigger a deterioration of GA performance. A requirement for the design on tightly linked codings is information about the BB structure – the decomposition – of a problem.

We propose a decomposition of the uWLP to gain insights into its BB structure. The decomposition is based on the idea to assign opening costs and relevant

transportation costs to open warehouses. Therefore, we split the set of warehouses into a set of open warehouses $W = \{W_1, \ldots, W_o\}$ whose size is denoted by $|W| = o$, and a remaining set of closed warehouses $\bar{W} = \{\bar{W}_1, \bar{W}_2, \ldots\}$. Closed warehouses are not assigned any costs directly. They enlarge the transportation costs associated with open warehouses. This is due to transportation requirements that accumulate at the open warehouses, if warehouses are being closed. We reformulate fitness function (3.1) as an additively decomposable function as follows.

$$\min \quad C = \sum_{k=1}^{o} f_{a_k} + \sum_{j \in A_k} c_{a_k,j}\, x_{a_k,j} \tag{3.4}$$

$$, \tag{3.5}$$

$$A_k = \cup_{j=1}^{N} \{\, j \mid x_{a_k,j} = 1\}, \tag{3.6}$$

$$B_j(k) = \cup_{i=1}^{M} \{\, i \mid c_{ij} < c_{a_k,j}\} \qquad \forall j \in A_k, \text{ and} \tag{3.7}$$

$$s_k = a_k \cup \left(\bigcup_{j \in A_k} B_j \right). \tag{3.8}$$

Just like in Section 3.2.1 we assign customers to their nearest open warehouse. o denotes the number of open warehouses in a solution. (3.4) is additively defined over o cost components that include opening costs and transportation costs that result from each of the opened warehouses. a_k denotes the index of the k-th open warehouse, f_{a_k} the associated opening costs. The index set A_k (3.6) comprises the customers that are assigned to warehouse a_k. Equation (3.7) returns for a customer $j = 1, \ldots, N$ that is assigned to warehouse a_k a set B_j that contains the indices of those warehouses that could have served customer j at lower cost than a_k. From the interaction of the set definitions and the assumption that all customers are assigned to their closest open warehouse follows directly that all warehouses $\bar{W}$ are closed. The set definition (3.8) defines the sets of decision variables s_k that directly or indirectly influence the costs caused by the open warehouse a_k.

We can construct the sets s_k for any instance of the uWLP if its optimal solution is given. The s_k can be regarded as the BBs of the warehouse location problem. They consist of a single bit that is 1 (the opening decision for warehouse a_k) and zeros for closed warehouses, if any. The opening decision causes opening costs and transportation costs from nearby customers. Spilled-over customers from closed warehouses increase transportation costs further.

We can now explain the numbering effect of Section 3.2.2. In Section 3.2.2, all sets s_k has been computed for each considered instance using the optimal solution. The worsening of GA performance was triggered by a numbering that spread bits belonging to the same s_k widely over the binary string. This was

achieved by setting them at equal distance. Thus, a coding with loose linkage was used. One-point crossover often disrupts high quality configurations of these bits and prevents them from appearing jointly in solutions. Larger population sizes are required for reliable convergence and thus, the number of fitness evaluations grows. The increase of GA performance and large savings in computation times were caused by choosing a numbering where bits that belong to the same s_k were coded closely together (tight linkage). The disruption of BBs is less likely in this case, and GA performance rises significantly.

3.2.4. Numerical study

Although the results of our initial example are startling, the question remains whether disruption of BBs is a relevant facet of GA behavior that optimization practitioners have to keep in mind when solving location problems. Disruption is clearly relevant if the sets s_k are highly overlapping and of large sizes in instances that are supposedly realistic. Then, any encoding that is chosen without respect to linkage information can be a reason for bad GA performance.

To answer this question we conducted an empirical analysis of 647 uWLP instances in 12 problem classes from the uncapacitated facility location library web page. The library comprises test instances from several sources and their associated optimal solutions. In cases where the optimal solution is not known, the best solution found so far is provided and was used in the analysis below. A short description of each problem class is available both on the UflLib web page and in Hoefer (2002). A pointer to the origin of each class, providing more detailed information on its nature, is given in column Org. of Table 3.2.

We generalized the classification scheme of Schilling et al. (2000) and distinguished instances according to how the transportation cost matrix was generated. *Euclidean* inter-point distances are based on a spatial representation in the plane where coordinates are known for each customer and warehouse location. *Network* distances are based on shortest paths between warehouses and customers on a random or predefined network such as road or railroad networks. *Random* distances are present if a distance matrix is generated randomly. Further, a distance matrix can be *full*, i.e., there is a true transportation cost value for each c_{ij}, or *sparse*, i.e., some connections are blocked by prohibitively large values.

The analysis of the instances has been conducted as follows. Starting from the optimal solution of each instance, the sets of interacting variables are computed according to equation (3.8). Then the sizes of the sets, the absolute and relative amount of decision variables that overlap with other sets, and the compliance of a solutions sets with the running intersection property was computed. Relative values are used to make instances of different sizes comparable. For each problem

class, the following characteristics are provided as average values over all instances in the class:

Class	Instance class name.
Org.	Pointer to source and/or description of problem class.
No.	Number of problem instances in the class.
Classification	Type of distance matrix.
Open	Percentage of potential locations opened in optimal solution.
Avg. SS	Average set size relative to total number of locations $[avg(\|s_k\|)/M]$.
Max. SS	Maximum set size relative to total number of locations $[max(\|s_k\|)/M]$.
Olp.	Overlap between sets as percentage of average set size.
RIP	Percentage of problems for which the running intersection property is fulfilled.

Table 3.2 presents the results. They show that overlapping and large sets of interacting variables are common in uWLPs. Optimal solutions are unknown in practical applications and a BB analysis as carried out in this chapter cannot be conducted. It follows that tightly linked codings cannot be designed in a principled manner. Under these circumstances, sGA do not seem to be the methods of choice because their efficiency is likely to be very sensitive to the numbering of the locations. With EDA, we propose an alternative in the next section.

Class	Org.	no.	Classification	Open	Avg. SS	Max. SS	Olp.	RIP
Bilde-Krarup	[1]	220	Random (full)	9.0%	87.5%	91.2%	63.2%	75.5%
Chessboard	[2]	30	Random (sparse)	11.1%	13.1%	16.1%	52.5%	0.0%
Euclidean	[2]	30	Euclidean (full)	13.7%	10.5%	17.3%	29.5%	6.7%
Finite PPs	[2]	59	Random (sparse)	6.8%	41.5%	54.9%	83.2%	0.0%
Galvao-Raggi	[3]	50	Network (full)	84.4%	2.5%	9.8%	16.2%	56.0%
Koerkel-Gosh	[4]	45	Random (full)	4.7%	56.7%	64.3%	82.0%	33.3%
k-Median	[5]	6	Euclidean (full)	1.1%	16.1%	20.9%	65.5%	0.0%
Duality Gap	[2]	90	Random (sparse)	13.6%	22.9%	36.6%	67.7%	0.0%
M*	[6]	15	Random (full)	2.5%	97.4%	98.1%	78.8%	100.0%
ORLIB	[7]	37	Network (full)	28.9%	17.6%	45.0%	17.7%	67.6%
	[8]	3	Biased Eucl. (full)	6.7%	45.1%	55.7%	63.0%	0.0%
PCodes	[2]	32	Network (sparse)	12.5%	15.6%	20.3%	60.3%	0.0%
Uniform	[2]	30	Random (full)	13.2%	19.4%	33.6%	61.8%	0.0%

Problem origins:

[1] Bilde and Krarup (1977)
[2] Kochetov and Ivanenko (2003)
[3] Galvão and Raggi (1989)
[4] Ghosh (2003)
[5] Barahona and Chudak (2005)
[6] Kratica et al. (2001)
[7] Akinc and Khumawala (1977)
[8] Beasley (1988)

Table 3.2.: Set characteristics obtained from popular uWLP test instances.

For EDA to work well, the set size $|b_i|$ and $|c_i|$ have to be polynomially bounded. The empirical study shows, that the sets s_k, which comprise both b_i and c_i is on average much smaller than the problem size.

As listed in Table 3.2, the RIP does not hold for most instances. Other decompositions without overlapping BBs could be constructed that would use larger sets and fulfill the RIP. However, compliance with the RIP is no necessary condition for an EDA to convergence to the global optimum and thus we refrain from doing so.

3.3. Experimental section

We perform experiments on Galvao-Raggi and ORLIB uWLP instances, as both are extensively used in benchmark studies. We use both a sGA and the Bayesian Optimization Algorithm. The BOA is able to identify the structure of decomposable problems and represents a state of the art multivariate EDA. Note that the EBNA or lFDA could have likewise been used. An experimental scalability analysis is conducted to assess how the average number of fitness evaluations required to solve the problems reliably grows with M.

3.3.1. Experimental design

Galvao-Raggi instances are based on randomly generated networks. The shortest path between warehouses and customers is used to obtain transportation costs. Warehouse opening costs are comparably low which leads to small set sizes and low overlap (see Table 3.2). The instances cover problem sizes from $N = M = l = 50$ to $N = M = l = 200$. ORLIB instances are widely used in the assessment of uWLP solution procedures. For the scale-up analysis, two subclasses have to be distinguished. The ORLIB instances with $M = 16, 25, 50$ are based on railroad distances between American cities, see Kuehn and Hamburger (1963). Instances of the same size differ with regard to warehouse opening costs. The instances with $M = 100$ are generated by randomly placing warehouses and customers in a 1000×1000 square, see Beasley (1988). Transportation costs are derived from Euclidean inter-point distances that are scaled using a random factor in [1,1.25].

The sGA uses tournament selection, bit-flip mutation, one-point crossover and generational replacement. Crossover and mutation probabilities were set to values that yielded best performance on the smallest instances of each problem class as follows. These probabilities may not sum up to 1.0. The probability of mutation is very low, whereas crossover is the major variation operator.

$M = N = l$	sGA	BOA	Speedup
50	8,625	4,457	1.94
70	88,200	7,604	11.60
100	168,273	14,437	11.65
150	600,085	19,956	30.07
200	941,390	29,175	32.27

Table 3.3.: Scalability results for the Galvao-Raggi uWLP instances.

	Galvao-Raggi	ORLIB
Tournament size	2	3
Probability of crossover	1.00	1.00
Probability of mutation	0.00	0.01

Mutation prohibits complete convergence in the sGA. A sGA run was terminated when, for each allele, the same value was present in 95% of the solutions. This criterion, however, was only applicable to the ORLIB instances. The Galvao-Raggi instances apparently feature multiple global optima. For these problems, the optimization run was terminated as soon as 95% of the individuals had converged to the same fitness value.

The BOA was set up to select the best 50% of each generation, from which its probabilistic model was built. The worse half of the population was replaced with newly generated individuals. A BOA run was terminated by bit-string convergence, that is all individuals are identical.

We made the following experiment for each algorithm and problem instance. The minimal population size required to solve the instance to optimality in at least 27 of 30 independent consecutive runs was determined by bisection. The number of fitness evaluations was averaged over successful trials. Results were averaged over instances of identical size.

3.3.2. Results and interpretation

Results of the scale-up analysis conducted on the Galvao-Raggi instances are provided in Figure 3.2 and Table 3.3. The values reported are averaged over the instances for each problem size. Both axes in Figure 3.2 have a logarithmic scale. Thus, straight lines represent polynomial scale-up behavior.

Figure 3.3 and Table 3.4 provide analogous information for ORLIB instances. The values reported are average values over four instances per size. On the $M = 100$ ORLIB instances, the sGA converged reliably for instance *capa* only.

Solution effort in number of evaluations e using the BOA has been determined to be $e \approx M^{1.33}$ and $e \approx M^{1.47}$ for the Galvao-Raggi and ORLIB instances

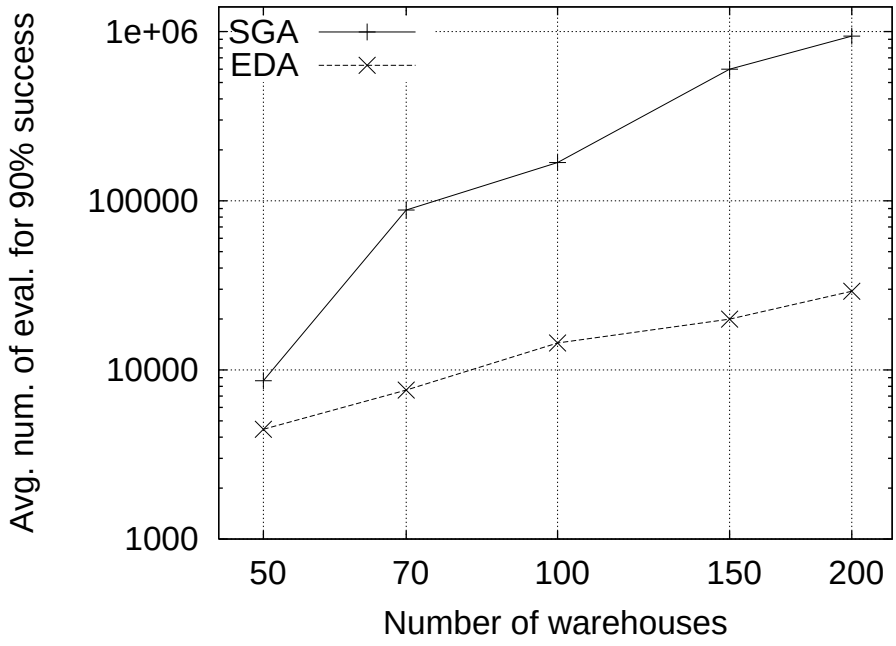


Figure 3.2.: Scalability results for the Galvao-Raggi uWLP instances. The graph plots the number of fitness evaluations necessary for a 90% success rate against M. Logarithmic scaling is used; straight lines indicate polynomial scalability.

$M = l$	N	sGA	BOA	Speedup
16	50	1,261	2,792	0.45
25	50	5,326	4,416	1.20
50	50	34,059	14,448	2.36
100	1000	305,679	28,473	10.74

Table 3.4.: Scalability results for the ORLIB uWLP instances.

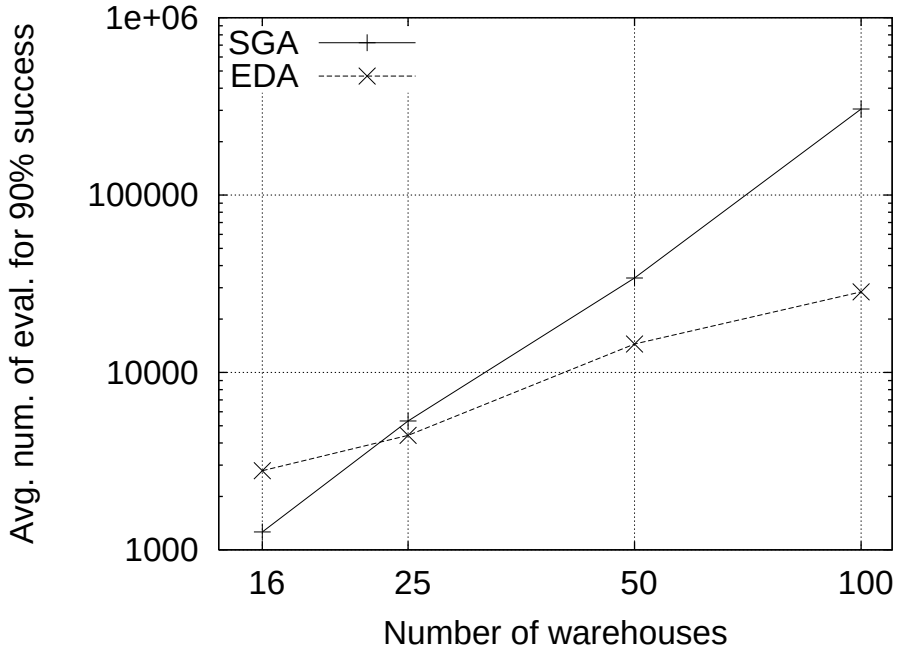


Figure 3.3.: Scalability results for the ORLIB uWLP instances. The graph plots the number of fitness evaluations necessary for a 90% success rate against M. Logarithmic scaling is used; straight lines indicate polynomial scalability.

respectively using regression. This is even lower than predictions from literature that predict a growth in between $O(M^{1.55})$ and $O(M^{2.05})$, see Pelikan et al. (2003). The good results might be attributed to the problems' favorable set structure. Both set sizes and overlap are comparably low for the problems considered. On average, the sGA's running time grows between $e = M^{2.87}$ (ORLIB) and $e \approx M^{3.20}$ (Galvao-Raggi), respectively. Again, this was approximated by regression.

From the results above, one can conclude that uWLPs are suited for both GA and EDA, but sGA are outperformed by EDA in terms of fitness evaluations. This is due to the inherent ability of EDA to learn linkage between warehouse opening decisions. Although the scalability of the BOA is better in terms of fitness evaluations, building large probabilistic models takes time. To decrease computation times that are caused by model building, various efficiency enhancements have been proposed in the literature, see Sastry et al. (2004).

Further, the experiments from Section 3.2.2 have been conducted using the BOA. Since variation in the BOA bases on density estimation and the structure of its probabilistic model is flexible with respect to the position of BBs, different numberings have no effect on its performance.

3.4. Summary and conclusion

Simple GA and EDA show good performance for solving uncapacitated warehouse location problems. We showed that the performance of sGA can nonetheless vary strongly and is not robust, depending on how opening decisions are mapped onto the genotype. It is difficult, if not impossible, to design tightly linked codings for an uWLP instance if its optimal solution is not known. The large speed-ups that are obtained from tightly linked codings make it worthwhile to investigate approaches that are capable of learning linkage when solving warehouse location problems.

We proposed to use EDA to solve uWLPs. EDA have been designed to solve additively decomposable functions with large and overlapping building blocks. EDA outperform GA and are insensitive to the warehouse numbering effect.

4. Solving safety stock allocation problems with evolutionary algorithms

Safety stock allocation problems in purely serial, divergent and convergent network topologies can be solved efficiently by dynamic programming. While the straightforward applicability of dynamic programs is limited to network topologies they have been designed for, dynamic programs for general networks suffer from the size and complexity of state and decision spaces. We challenge the paradigm of designing problem-specific algorithms and report on the reliability and runtime of evolutionary black-box optimizers on safety stock problems. They exploit little problem specific knowledge but expected total inventory holding costs of a safety stock allocation that results from a well-known extreme point property. This reduces the costs of algorithm design to a minimum. The contribution of this chapter is to indicate that evolutionary algorithms are capable of routinely solving safety stock allocation problems in different topologies to global optimality. Superior topology-algorithm matchings are recommended on the basis of an experimental study.

4.1. Introduction

Multi-echelon inventory control is one area of supply chain management that has attracted ample research during the last decades. Early contributions to multi-echelon inventory theory focused on analytical insights and were restricted to rather simple settings like two stages, identical retailers, negligible lead times, to name just a few. For a general overview of the development we refer to the reviews by Diks et al. (1996), van Houtum et al. (1996) and Axsäter (2003).

Recently, more emphasis has been put on the development of methods for determining inventory levels and control parameters in networks of more general structure and under less restrictive assumptions with regard to materials flow and information requirements. According to van Houtum et al. (1996) and Graves and Willems (2003), these models fall into two major categories: full-delay/stochastic-service models and no-delay/guaranteed-service models. The approaches differ in terms of the underlying materials flow concept and the resulting service-time characteristics. The service time is the time it takes until materials ordered by a stock-point are made available for processing by its predecessor.

The no-delay approach with its fundamental work by Simpson (1958) and the

underlying base-stock concept proposed by Kimball (1988) assumes deterministic service times. Here, orders of any size can be met by the supplying stage within the quoted service time. This is enabled by the assumption that, in case of insufficient inventory to meet demand, extraordinary measures, e.g. overtime or accelerated production, exist, so-called operating flexibility, by which the missing amount is made available in time. This allows for a decomposition of the overall optimization problem into a master coverage allocation problem and individual safety stock sizing problems for each stage given the allocation coverage requirement. Assuming independent normally distributed demands the optimization problem has the special structure of a concave minimization problem subject to linear constraints for which an extreme point property holds (see e.g. Horst and Tuy (1998)).

Solution techniques comprise general purpose concave minimization algorithms (see e.g. Horst and Tuy (1998)) and methods that make explicit use of an extreme point property. Here, dynamic programming is the most prominent one. For divergent networks Minner (1997) presents backward recursion formulations. Similarly, convergent networks can be solved by forward recursion formulations (see Minner (1997)). An extension of these basic network types to more realistic topologies is presented by Graves and Willems (2000). A single state variable dynamic programming algorithm is developed for spanning trees and is further improved by Lesnaia (2004). Humair and Willems (2006) build on the result by Graves and Willems (2000) and extend it to so-called clusters of commonality networks where component commonality occurs only between adjacent echelons. This approach is applicable to fairly general network types and comprises pure topologies. Lesnaia (2004) shows that the general safety stock coverage problem is NP-hard, characterizes candidates for an optimal solution and uses these properties to develop a branch-and-bound algorithm. For general acyclic networks, Minner et al. (2006) present a dynamic programming approach.

Besides solution techniques that guarantee optimality, several heuristic methods have been proposed in the literature. Minner (2000) presents several local search algorithms, e.g. simulated annealing, threshold accepting and tabu search. Magnanti et al. (2006) approximate the concave objective function by piecewise linear functions, thus enabling the use of standard MIP solver technology to find the optimal solution. The authors state that even networks of general acyclic type can be solved within reasonable computation time, but detailed information on runtimes is missing.

We are interested in the applicability of global *black-box* optimization to solve safety stock problems. We considered a stochastic hill-climber with the (1+1)-EA , the simple genetic algorithm and the Bayesian optimization algorithm, a state-of-the-art estimation of distribution algorithm. These methods embody different schools of thought in black-box optimization. They are widely available, well understood, and require only a minor modification for solving safety stock problems

in different network types: the objective function has to be implemented and a solution representation must be chosen. No modification is made to the algorithms' search strategies. This dramatically reduces the complexity of algorithm design compared to topology-specific algorithms. However, stochastic heuristics can not guarantee global optimality of the best found solution. Furthermore, a seperated investigation of different problem topologies facilitates the understand of the behaviour of the algorithms although complex networks comprise the pure network types that we study as well.

Therefore, the central question of this work is how reliable in terms of success ratios the evolutionary approaches are when solving safety stock allocation problems in serial, divergent, convergent and general acyclic supply networks. Further, we are interested in their running times.

Section 4.2 reviews a mathematical model of the safety stock allocation problem in general acyclic supply networks. Section 4.3 briefly reviews the evolutionary algorithms which are used in Section 4.5 to solve the safety stock problems. The chapter ends with conclusions and an outlook on future research. A preliminary version of the chapter is available in Dittmar et al. (2007).

4.2. The guaranteed service time safety stock allocation problem

We make frequent use of mathematical symbols and some specific notation, all of which are introduced in the following. Single echelons i in a supply network consisting of $i = 1, 2, \ldots, n$ nodes have immediate predecessor nodes $v(i)$ and immediate successor nodes $n(i)$. The terms node, product and stock-point are used interchangeably. All nodes with external supply and no predecessor nodes are in set A, all intermediate nodes are in set P. End-item nodes without successor nodes that face customer demand are in set E.

Demand occurs at nodes i in E and has expected single period demand μ_i. The standard deviation of single period demand for product i is σ_i. The coefficient of correlation between single period demand for products i and j is ρ_{ij}.

Inventory holding costs per unit and unit of time at stock-point i are h_i. The processing time for product i is λ_i. Stock-point is service level is SL_i. The customer service time for nodes $e \in E$ is $\bar{ST}_e$. Furthermore, e_i denotes the excess coverage provided by downstream stock-points which is available at stock-point i.

The supply network is modeled as a directed, acyclic graph G. Each node performs a certain processing function, e.g. a manufacturing or transportation process, and is a potential location for holding safety stock after the process has

finished. The arcs (i, j) indicate material flow requirements between stock-points i and j. The set A contains all stock-points i that have no immediate predecessors $v(i)$ in G, i.e. $v(i) = \emptyset \ \forall i \in A$, but procure raw material from external sources. Stock-points with at least a single predecessor and successor in G are assigned to set P. All stock-points that directly face stochastic end-product demand and therefore have no immediate successors $n(i)$ in G, i.e. $n(i) = \emptyset \ \forall i \in E$, are in set E.

End-product demand at stock-point i is assumed to be normally distributed with mean μ_i and standard deviation σ_i. Unsatisfied customer demands are back-ordered.

Dependent internal demand at stock-point i results from the input coefficients specified in the bill of materials (BOM). These coefficients are assumed to be 1 throughout the chapter. Each stock-point operates a periodic review base-stock policy with the review period being identical at all stock-points. Consequently, every review triggers a material request at all supplying stock-points for exactly the difference between the base-stock level of a stock-point and its inventory position (i.e. outstanding orders plus on-hand inventory minus back-orders.).

The processing time at each stock-point λ_i is assumed to be deterministic and an integer multiple of the review period. λ_i denotes the time required for the transformation process at node i, given that all required input materials are available. Whether materials are available immediately or after a certain period of time depends on the so-called service times of the preceding stock-points, $ST_{v(i)}$. A service time of 0 corresponds to immediate material availability. Furthermore, it is assumed that each stock-point quotes the same service time to all of its successors. Due to the coupling of adjacent stock-points via service times, the time span for which protection against demand uncertainty is required is given by

$$T_i = \max_{j \in v(i)} \{ST_j\} + \lambda_i - ST_i \ . \tag{4.1}$$

The coverage time T_i consists of the replenishment lead time of stock-point i, $\max_{j \in v(i)}\{ST_j\} + \lambda_i$, minus coverage requirements postponed to successors through a (positive) service ST_i. The coverage time calculation ignores the impact of potential stockouts at supplying stock-points and assumes deterministic predecessor service times. The corresponding amount of safety stock can be determined by

$$SST_i = k_i(SL_i)\sigma_i\sqrt{T_i} \quad . \tag{4.2}$$

SST_i is analogous to the well-known square root safety stock formula of the single-echelon case where k_i is a so-called safety factor that depends on the service level SL_i. The service level is assumed to be of the *alpha*-type in the experimental section, that is the probability of experiencing a stockout at the end of a period is constrained by $(1 - \alpha)$. Assuming that demand is normal, k_i is computed from

the inverse of the normal cumulative distribution function.

The resulting safety stock allocation optimization problem for an arbitrary number of stock-points n can be stated as follows

$$\min C = \sum_{i=1}^{n} h_i k_i(SL_i)\sigma_i \sqrt{\max_{j\in v(i)}\{ST_j\} + \lambda_i - ST_i} \tag{4.3}$$

$$s.t. \quad 0 \leq ST_i \leq \max_{j\in v(i)}\{ST_j\} + \lambda_i \qquad \forall i \notin E \tag{4.4}$$

$$0 \leq ST_i \leq \overline{ST}_e \qquad \forall i \in E \tag{4.5}$$

The goal of the optimization is to find the service time combination and in turn safety stock levels that minimize total safety stock holding costs in the supply network. Note that in the upper formula, the safety stock is used as an approximation for the on-hand stock. Following Silver et al. (1998), this approximation is feasible due to a negligible chance of experiencing backorders when high external service levels like 95% are assumed and internal service levels are set equal to the external ones. This follows the relevant literature.

Constraints (4.4) ensure a positive service time at each stock-point and further restrict the maximum service time to the stock-point's replenishment lead time. Constraints (4.5) represents a special case of (4.4) for stock-points facing end-product demand. The service time $\overline{ST}_e$ is a parameter rather than a decision variable and specifies the acceptable customer waiting time.

Under normally distributed demand, service levels of the α-type and a serial network topology the objective function is non-linear and concave. For complex network topologies, concavity is used as an assumption. Note that an extreme point property holds, that is, an optimal service time (coverage) allocation is obtained on the edges of the feasible region, see, e.g., Horst and Tuy (1998) and Magnanti et al. (2006) for a linearization of the max operator. This property implies, that a stock-point either covers zero time (and therefore postpones the coverage of its replenishment lead time to its direct successor locations) or holds sufficient safety stock to cover its entire replenishment lead time, see Simpson (1958) and Minner (2000).

Whereas in serial and divergent networks, possible coverage time candidates only comprise cumulative processing time values, in convergent and general networks additional solution candidates have to be considered resulting from so-called excess coverage values $M(i)$. Excess coverage may occur at convergent substructures in the network. Stock-point i experiences an excess coverage time of $M(i)$, if the service time that stock-point i quotes to its immediate successor k is smaller than the service time of another stock-point j which has the same immediate successor k. In case stock-point k decides to hold safety stock it has to cover the longer replenishment lead time, i.e. $ST(j) + \lambda_k$, and therefore stock-point i

receives excess coverage of $M(i) = ST_j - ST_i$.

4.3. Representation and (1+1)-EA

The (1+1)-EA, the simple genetic algorithm, and an estimation of distribution algorithm have been chosen for comparison. These are well understood black box optimization routines and representative for different variation paradigms encountered in EC. The (1+1)-EA uses a *single* solution as input for variation. Variation is performed through randomly mutating parts of the solution. In contrast, the simple genetic algorithms is population-based. It takes *two* solutions, called parents, as inputs for variation and employs crossover-type recombination operators to combine bits and pieces of the parents. Mutation is performed with low probability and crossover is the dominating operator. The estimation of distribution algorithm tries to learn problem structure by estimating a probability density from *all* selected solutions. This density is sampled from to generate offspring. This section describes the solution representation used by the EA and briefly introduces the (1+1)-EA.

A core element of evolutionary algorithms such as the (1+1)-EA, the simple genetic algorithm and the estimation of distribution algorithm is a solution representation which defines a mapping between actual solutions to the problem at hand (the phenotype) and a string of solutions components Z (the genotype), see Rothlauf (2006). Whereas the quality of a solution is measured on the phenotypic level, new solutions are constructed by variation steps that modify the genotypes.

Following the representation introduced in Minner (2000), it is sufficient to encode the stock-points coverage decisions as binary values. 1 denotes that a stock point holds inventory, 0 denotes a stock-point that does not. This binary representation is unique and sufficient in order to represent all possibly optimal solutions of safety stock allocation problems in serial and divergent networks. Note that the mapping between genotypes and phenotypes is surjective. A single solutions can be decoded by more than a single genotype, but all feasible phenotypes can be decoded on binary strings. For convergent and general acyclic networks we follow the mainstream assumption that optimal solutions can equivalently be characterized. The coverage decisions are used along with the network topology in a decoding procedure to obtain coverage times which in turn are evaluated with holding costs.

Thus, a phenotype is a safety stock allocation in the network. It is represented by a genotype that is a binary string of fixed length. In serial networks the coverage decision of node $i = 1, 2 \ldots, n-1$ is mapped onto bit i of the genotype, see Figure 4.1 for an example. All nodes that face end-customer demand hold safety stock. Regardless of the network topology, these nodes are not encoded on the binary

string because their genotypic values will always be 1. Thus, the problem size $|\mathsf{Z}| = l$ differs from the number of nodes as $|\mathsf{Z}| = l = n - |E|$.

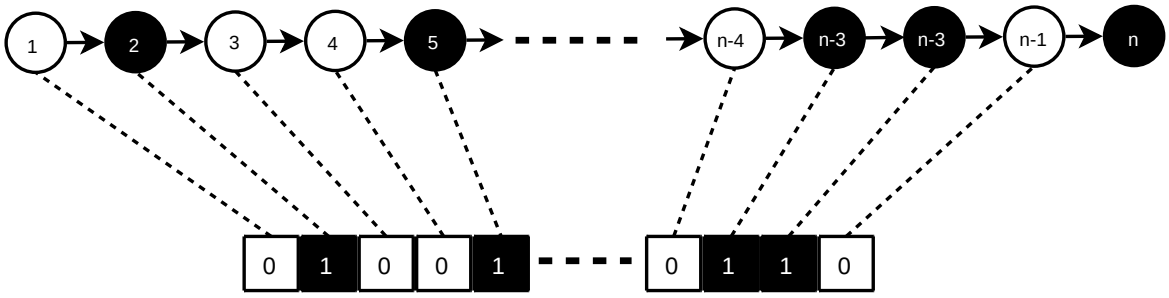


Figure 4.1.: Representation of solutions in a serial network. Black nodes represent stock-points that hold safety stock.

We use a mapping illustrated in Figure 4.2. Starting from the most upstream stock-point the network is traversed stage-wise. Nodes from the same stage are encoded next to each other starting with the topmost node in a graphical illustration to the bottommost node. The same encoding is used for serial, divergent, and convergent topologies. As illustrated in Chapter 3, the positioning of decision variables on the bitstring can have a significant impact on the performance of genetic algorithms. Yet, an analysis of linkage in safety stock allocation problems is beyond the scope of this chapter.

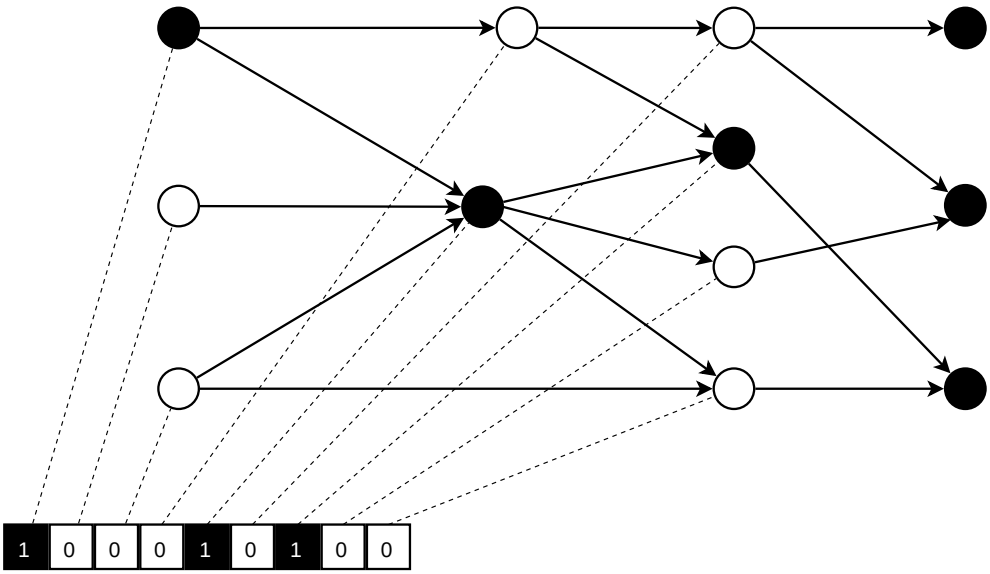


Figure 4.2.: Representation of solutions in a general acyclic network. Black nodes represent stock-points that hold safety stock

The coverage times are compiled from a binary string Z using a two-stage algorithm proposed by Minner (2000). Optimal coverage times T_i can be characterized

by

$$T_i = (L(i) - M(i))^+ \qquad i = 1, 2, \ldots, n, \tag{4.6}$$

with replenishment lead times $L(i)$ and excess coverage times $M(i)$. A binary vector encodes a single, well defined set of coverage times T_i. Note that $L(i)$ can be evaluated independent of $M(i)$. In a first step the replenishment lead times $L(i)$ are computed, then excess values $M(i)$ are derived.

To obtain replenishment lead times $L(i)$, the network is traversed starting from nodes in A. For these nodes $i \in A$, the replenishment lead time $L(i)$ is λ_i. The remaining nodes are visited in a topological order. Each node is visited, after all its predecessors have been visited already. If a predecessor j to i holds safety stock ($\mathsf{Z}(j) = 1$), its service time is zero and does not influence node i's replenishment lead time. The effect of λ_j is zero, it is thus multiplied by $(1 - \mathsf{Z}(j)) = 0$. If a predecessing node j does not hold safety stock, $L(j)$ might influence the replenishment lead time of node i and is multiplied by $(1-\mathsf{Z}(j)) = 1$ for potential cumulation. Formally,

$$L(i) = \lambda_i \qquad \forall i \in A, \tag{4.7}$$

$$L(i) = \lambda_i + \max_{j \in v(i)} \{L(j) \cdot (1 - \mathsf{Z}(j))\} \qquad \forall i \in P \cup E. \tag{4.8}$$

Excess values $M(i)$ are deduced starting with stock-points without successors. The excess of these nodes $i \in E$ is zero by definition. All other nodes are visited in a sequence where a node is visited when the coverage time values for all of its successors have already been derived. Formally,

$$M(i) = \bar{ST}_i - \lambda_i \qquad \forall i \in E, \tag{4.9}$$

$$M(i) = \min_{j \in n(i)} \{T_j - \lambda_j + M(j)\}^+ \qquad \forall i \in A \cup P. \tag{4.10}$$

Having derived replenishment lead times and excess coverage, the costs associated with Z are

$$C(\mathsf{Z}) = \sum_{i=1}^{n} h_i \cdot \sigma_i \cdot k_i \cdot \sqrt{T_i(\mathsf{Z})} \tag{4.11}$$

and are used in the algorithms to evaluate the expected costs of safety stock in the solution expressed by Z.

The (1+1)-EA is the simplest EA that can be designed. It is a stochastic hill-climbing algorithm that searches inside a bit-flip neighborhood as follows. An initial binary solution vector is generated randomly. In a variation step, each bit is considered for being flipped with an exogenously chosen mutation probability. The modified solution is the offspring. If its quality is superior to that of its parent, the offsprint replaces the parent, and is deleted otherwise. The (1+1)-EA iterates the mutation and replacement steps until a stopping criterion is met.

4.4. Decomposition of serial safety stock allocation problems

Using the binary representation introduced in Section 4.3, this section demonstrates separability of the serial safety stock allocation problem. Separability is demonstrated by formulating cost function (4.11) as an ADF. The set of n binary decision variables is partitioned into m disjoint sets $s_i, i = 1, \ldots, m$. Each set holds part of all indices for stock-points $1, 2, \ldots, n$, such that $s_i \subseteq \{1, 2, \ldots, n\}$ and $s_i \cap s_j = \emptyset \forall i \neq j$. Furthermore, the sets s_i contain adjacent indices. Total cost is expressed by

$$C = \sum_{i=1}^{m} \left[\sum_{j \in s_j} \mathsf{Z}_j h_j \sigma_j k_j \sqrt{\sum_{k \in s_i} \lambda_k} \right]. \tag{4.12}$$

Cost function (4.12) decomposes the overal cost calculation into m terms. Each term expresses safety stock holding costs that arise in the $i-$th part of the serial network, $i = 1, 2, \ldots, m$. The m parts are non-overlapping and obtained as follows. In total, $\sum_{i=1}^{n} \mathsf{Z}_i = Q$ binary variables Z_i are 1. Q nodes hold safety stock. A vector $\boldsymbol{q}' = (q_1, \ldots, q_Q)$ is introduced that contains all indices of exactly these nodes in increasing order, that is $q_1 = \min\{q_j, j = 1, \ldots, Q\}$. Based on this information, the sets s_i are constructed as

$$s_i = \begin{cases} \bigcup_{k=1}^{q_1} k & i = 1, \\ \bigcup_{k=q_{(i-1)}+1}^{q_i} k & i > 1. \end{cases} \tag{4.13}$$

By (4.13) the s_i contain a single index that is associated with a binary variable that is 1 (one per set s_i). All other indices in a set (if any exist) link to a binary variable that is 0. These adjacend nodes lie more upstream than the node that holds safety stock. The resulting cost function is identical to (4.11) and an ADF, since the sets s_i do not overlap.

This result shows that selector-recombinatorial optimization techniques like GA and EDA are in principal suited to solving the safety stock allocation problems in serial topologies. In divergent systems, a similar decomposition should apply. Convergent and general acyclic topologies result in potential excess coverage values that complicate the dependencies between decision variables.

Since the representation used in this chapter aligns decision variables that belong to a dependency set on the binary string (for serial problems), it proposes a tightly linked coding that should be especially suited for GA. In more complex network types, renumbering effects like the one illustrated in Chapter 3 are likely to have an impact on the performance of GA. An analysis of the dependency structure and numbering effects in complex networks is future work.

4.5. Experimental study

How effective and efficient are the above black-box optimizers on safety stock allocation problems? To answer this question, we performed experiments on serial, divergent, convergent and general acyclic supply networks. The experimental setup is similar for all network topologies. We consider both small and larger networks for each topology. 20 instances of safety stock problems are randomly generated for each size and all optimal solutions are obtained from dynamic programs (Minner (2000), Minner et al. (2006)).

A critical parameter that needs to be set when using the simple GA and the EDA is the population size. Too small population sizes hinder convergence, while too large populations waste computational resources. In order to obtain insights into the appropriate choice and impact of population sizes, we varied them systematically. In a first step, each problem is solved with a population size that equals the problem size. The problem size is the number of nodes that do not face end-customer demand. A population size is used in 10 independent trials for each of the 20 instances. Termination criteria are identical for the GA and the EDA. A trial is terminated if the optimal solution has been found, or the best found solution does not improve over 100 generations.

The success ratio is the percentage of runs in which the optimal solution has been found by an algorithm. It is averaged over all instances and trials. Additionally, the total number of fitness evaluations is recorded for trials in which the optimal solution has been found. It is averaged over all successful trials. The average number of fitness evaluations serves as a hardware-independent measure of performance. The population size is doubled in subsequent steps until it reaches a maximal value of 10.000 individuals or until the success ratio is larger than 80%. The simple GA uses one-point crossover and a mutation probability of $1/l$. Truncation selection selects 50% of the individuals in both the GA and the EDA.

The (1+1)-EA does not employ a population of individuals. The maximum number of fitness evaluations required by the GA or the EDA for a given population size is used as the maximum allowed number of evaluations for a single run of the (1+1)-EA. Its mutation probability is set to $1/l$. The (1+1)-EA is terminated after 5000 non-improving evaluations.

The experiments result in an empirical distribution of the success ratios and number of evaluations given different population sizes are obtained. This distribution allows for a comparison of the algorithms with respect to reliability and runtime.

The following problem-specific parameters are used identically for all topologies. End-customer demand is normally distributed with mean 10 and variance 9 and uncorrelated between stock-points. A 95% service-level of the α-type is enforced in all stock-points, resulting in safety factors $k_i = 1.65$. Customer service times

$\bar{s}_e$ for end-item stock-points are 0. Processing times λ_i are uniformly distributed in $[1:5]$. Holding costs h_i at a stock-point are

$$h_i = \frac{z_i}{\sigma_i \cdot k_i}. \tag{4.14}$$

The parameter z_i is set differently across topologies and instances as explained in the following sections. The general idea is to set z_i such that optimal solutions are stimulated, that the instances have a value-added cost structure that reflects value-adding transformation processes that occur at nodes. Further, the optimal solutions should contain safety stock at nodes in A and P. The experimental results obtained for the stochastic hillclimber indicate that the instances are multimodal for a search operator that flips a single bit.

In order to obtain such properties, an indifference ratio is obtained in a first step. If z_i is set to this ratio, the safety stock holding costs that are caused by holding safety stock at stock-point i are equivalent to those caused by not holding safety stock at stock-point i. In a second step, random numbers are sampled that can be higher and lower than the indifference ratio, resulting in solutions where coverage decisions are distributed over the network. It has been checked, that this methodology generates multimodal serial instances. It is conjectured, that multimodality also holds for the other network types.

Consider a two-echelon serial toplogy with nodes 1 and 2. Node 2 faces end-customer demand and holds safety stock. Node 1 supplies node 2 and might hold safety stock, if this reduces total safety stock holding costs in the system. The two decisions ($\mathsf{Z}_1 = 0/1$) cause identical total costs, if

$$h_1\sigma_1 k_1\sqrt{\lambda_1} + h_2\sigma_2 k_2\sqrt{\lambda_2} = h_2\sigma_2 k_2\sqrt{\lambda_1 + \lambda_2}. \tag{4.15}$$

Assuming identical standard deviation of demand and safety factors across stock points and simplyfying leads to

$$h_2 = h_1 \cdot \frac{\sqrt{\lambda_1}}{\sqrt{\lambda_1 + \lambda_2} - \sqrt{\lambda_2}}. \tag{4.16}$$

We generalize (4.16) to n nodes in a serial network where node i holds all safety stock and an additional coverage decision of node $i-1$ has no impact on total

costs:

$$h_{i-1}\sqrt{\sum_{k=1}^{i-1}\lambda_k} + h_i\sqrt{\lambda_i} = h_i\sqrt{\sum_{k=1}^{i}\lambda_k} \tag{4.17}$$

$$\Leftrightarrow h_i = h_{i-1}\cdot\left[\frac{\sqrt{\sum_{k=1}^{i-1}\lambda_k}}{\sqrt{\sum_{k=1}^{i}\lambda_k}-\sqrt{\lambda_i}}\right]. \tag{4.18}$$

By randomly sampling holding costs around h_i, randomized optimal coverage patterns are stimulated. Note that setting holding costs as in (4.18) leads to a value-added holding costs structure because $h_i \geq h_{i-1}$:

$$h_i \geq h_{i-1} \tag{4.19}$$

$$\Leftrightarrow \left[\frac{\sqrt{\sum_{k=1}^{i-1}\lambda_k}}{\sqrt{\sum_{k=1}^{i}\lambda_k}-\sqrt{\lambda_i}}\right] \geq 1 \tag{4.20}$$

$$\Leftrightarrow 2\left[\sqrt{\lambda_i\sum_{k=1}^{i-1}\lambda_k}\right] \geq 0. \tag{4.21}$$

Since all $\lambda_i \geq 0$, (4.21) and thus (4.19) must be true. This idea is generalized to divergent and convergent problem structures in the following sections.

4.5.1. Serial topology

In a serial network, the most upstream stock-point has an index of $i = 1$. Parameter $z_1 = 1$, and $z_i, i = 2, \ldots, n$ is set to

$$z_i = z_{i-1}\cdot U\left[1; 2\cdot\frac{\sqrt{\sum_{k=1}^{i-1}\lambda_k}}{\sqrt{\sum_{k=1}^{i}\lambda_k}-\sqrt{\lambda_i}}\right], \tag{4.22}$$

where $U[a;b]$ denotes a realization of a random variable that is uniformly distributed in $[a;b]$. Note, that $a = 1$, and b is set to twice the indifference ratio that has been described above.

The experimental results are illustrated in Figure 4.3 for small networks of 10 stock-points. The results for larger supply chains with 25 stock-points are presented in Figure 4.4. The horizontal axes list population sizes in a logarithmic

scaling. The vertical axes list average success ratios and average runtimes associated with these population sizes. Missing points indicate that the algorithm was unable to at least once solve an instance to optimality given this population size.

It is obvious that the generated instances do not pose problems for either evolutionary algorithm given the population size is high enough. Given a fixed population size, the sGA attains the highest success ratio. It is important to note that it also requires the highest runtimes for any population size. The runtime results suggest that the (1+1)-EA is the most efficient algorithm out of the selected ones for solving the instances. This might be due to the underlying simple structure of serial safety stock allocation problems. Nodes that hold safety stock decouple the supply chain. This logic limits interaction between decision variables. As a result, flipping single bits in a (1+1)-EA suffices to solve the instances to optimality.

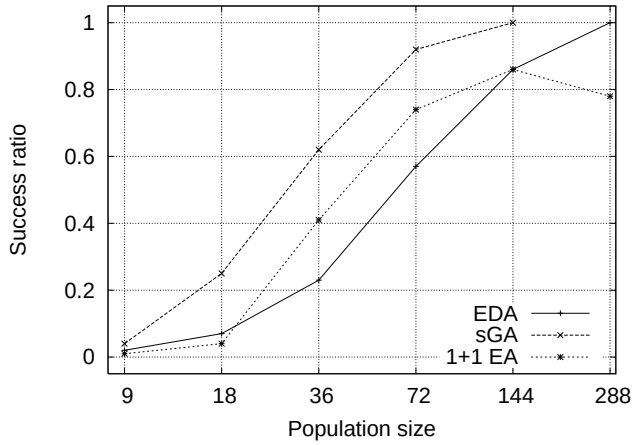
4.5.2. Divergent topology

In divergent topologies, the most upstream stock-point has an index $i = 1$. The parameter z_i is set to

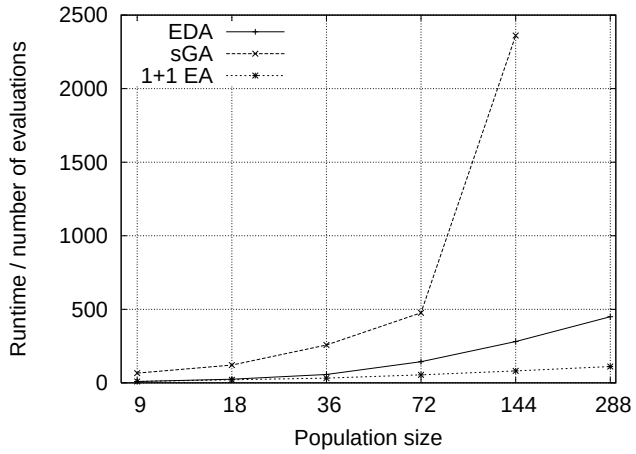
$$z_i = 100 \forall i \in E, \text{ and} \tag{4.23}$$

$$z_i = \sum_{j \in n(i)} z_j \cdot U \left[1; \frac{\sqrt{\sum_{k \in V(j)} \lambda_k} - \sqrt{\lambda_j}}{2 \cdot \sqrt{\sum_{k \in V(i)} \lambda_k}} \right]. \tag{4.24}$$

The network structure is depicted in Figure 4.5(a) for problems with 28 nodes. Small networks have the same triangular structure but less nodes. The results depicted in Figure 4.6 are similar to the ones obtained for serial networks. This is in line with the structural similarities between serial and divergent safety stock problems. For either size no clear distinction can be made between the success ratios of the algorithms. Given that populations are large enough (resp. the maximum runtime for the (1+1)-EA), the EA solve the instances reliably to the optimum. The sGA again requires the highest number of fitness evaluations. Optimization speed renders the (1+1)-EA a sensible choice for solving safety stock allocation problems in small divergent networks. For larger networks, Figure 4.7 presents different results. The (1+1)-EA is incapable of solving these instances with required reliability. The inherent functionality of the (1+1)-EA suggests that this is due to getting trapped in a local optima. Obviously, the number of local optima increases faster than the problem size, rendering larger instances harder to solve. The EDA proves to be more reliable than the GA. At the same time, it reaches the optimum faster.

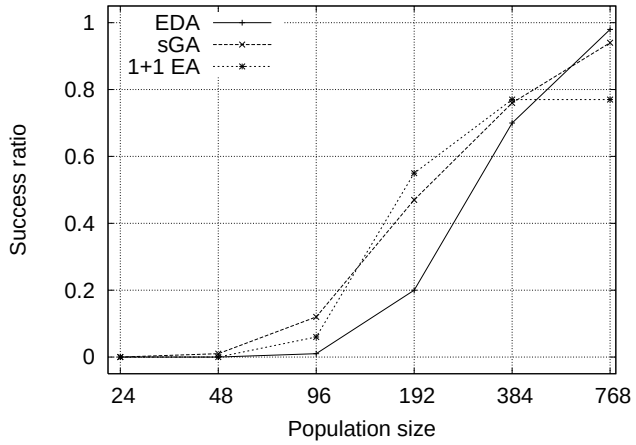


(a) Success ratios

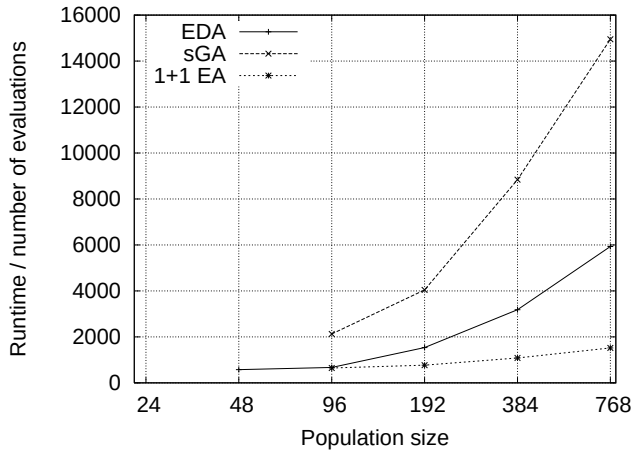


(b) Runtime/evaluations

Figure 4.3.: Success ratios and runtime for serial networks with 10 stock-points

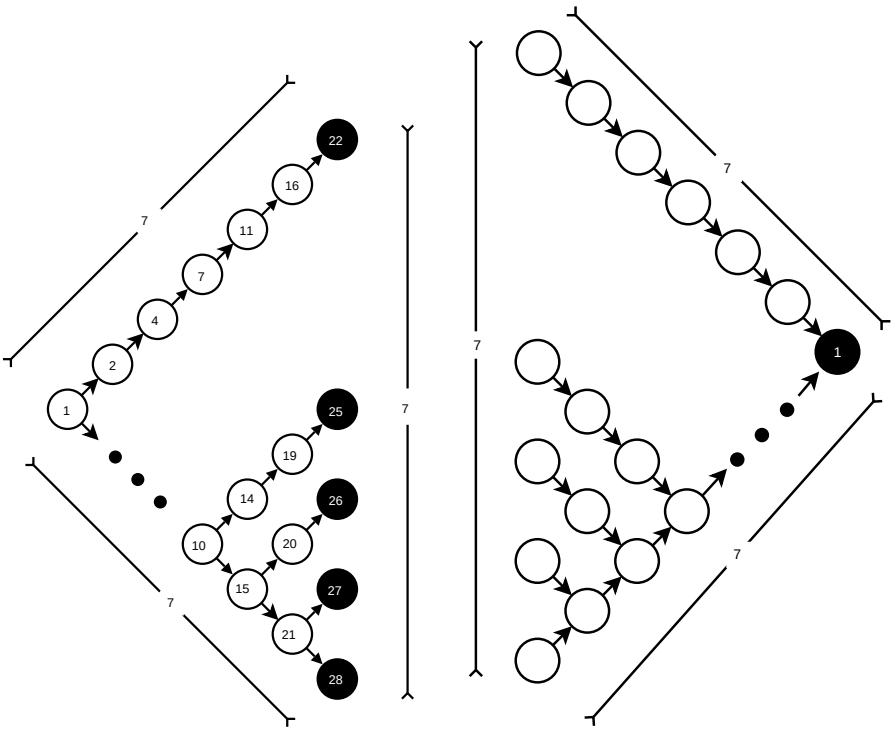


(a) Success ratios



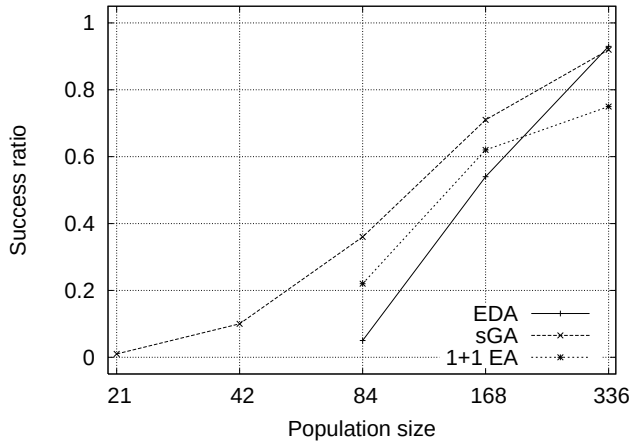
(b) Runtime/evaluations

Figure 4.4.: Success ratios and runtime for serial networks with 25 stock-points

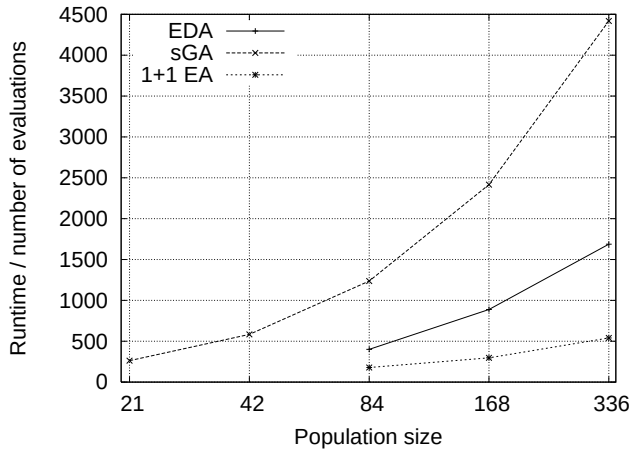


(a) Divergent network (b) Convergent network

Figure 4.5.: Network topology for large divergent and convergent networks.



(a) Success ratios



(b) Runtime/evaluations

Figure 4.6.: Success ratios and runtime for divergent networks with 28 stock-points

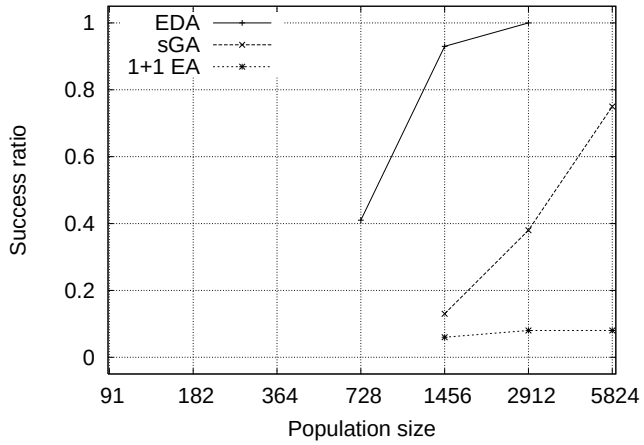
4.5.3. Convergent topology

In convergent networks the most upstream stock-point has index $i = 1$ and z_i is set to

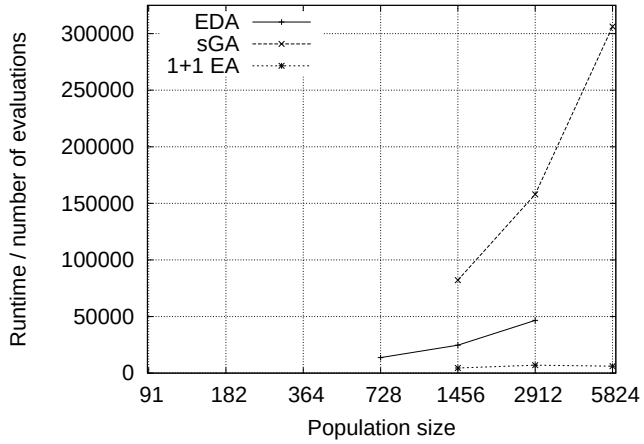
$$z_i = 1 \forall i \in A, \text{and} \tag{4.25}$$

$$z_i = \sum_{j \in v(i)} z_j \cdot U \left[1; 2 \cdot \frac{\sqrt{\max\limits_{w(l,j) \in W(l,j), l \in A} \sum_{k \in w(l,j)} \lambda_k}}{\sqrt{\max\limits_{w(l,i) \in W(l,i), l \in A} \sum_{k \in w(l,i)} \lambda_k} - \sqrt{\lambda_i}} \right]. \tag{4.26}$$

The network topology is illustrated in Figure 4.5(b). The results from the ex-



(a) Success ratios

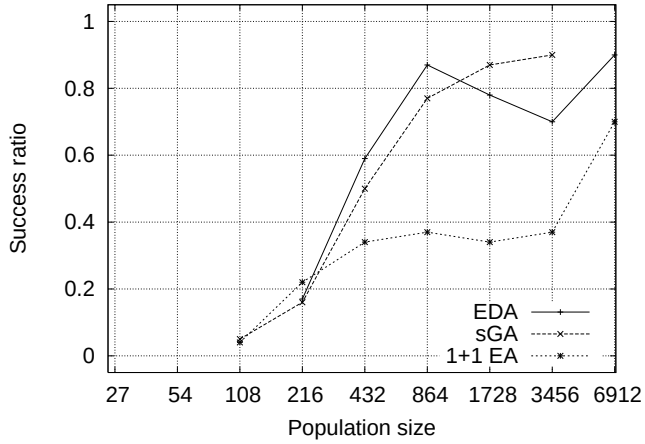


(b) Runtime/evaluations

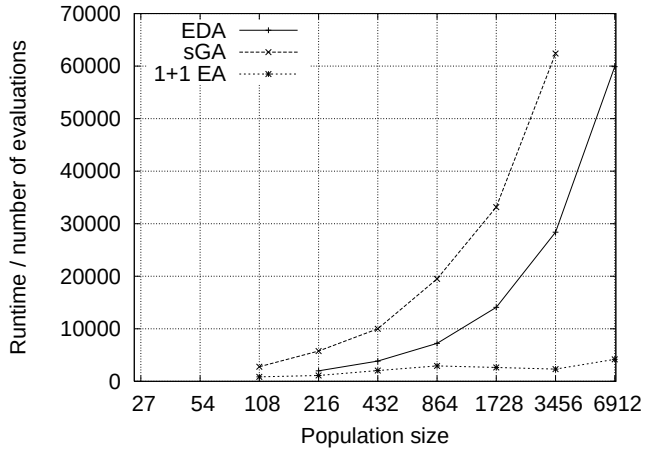
Figure 4.7.: Success ratios and runtime for divergent networks with 105 stock-points

periments are graphed in Figures 4.8 and 4.9. It can be seen that the (1+1)-EA is not able to reliably solve neither small nor large instances. We suggest that this is due to a higher number of local minima. Further, excess coverage times and their influence reduce the effectiveness of simple bit flipping strategies as the complexity and strength of interactions between coverage times and coverage decisions between stock-points increase. The sGA and the EDA are both capable of solving the small instances reliably, but runtime results are in favor of the EDA. Considering the large instances, however, a significant deterioration of performance is witnessed for the sGA. It is not able to solve the instances to optimality in a reliable manner. In contrast, the EDA manages to generate optimal solutions with desired reliability. At the same time, its running times are considerably lower than those of the sGA. Note that the EDA requires much

smaller populations to generate optimal solutions at all.



(a) Success ratios

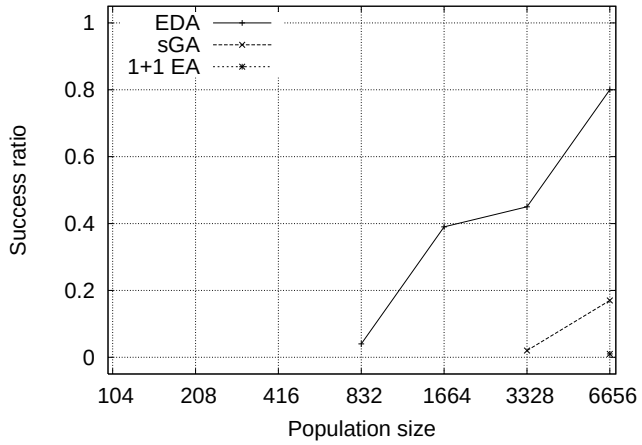


(b) Runtime/evaluations

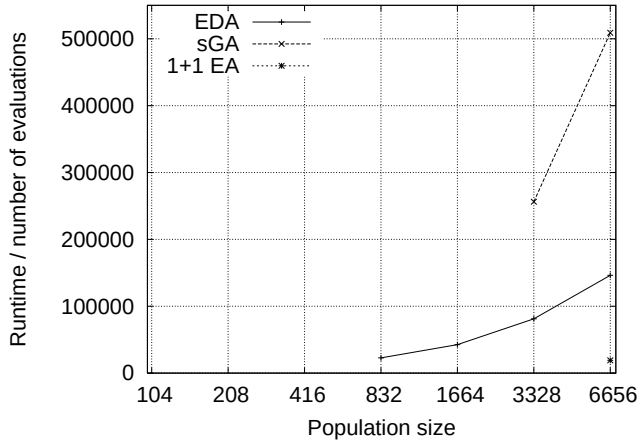
Figure 4.8.: Success ratios and runtime for convergent networks with 28 stock-points

4.5.4. General acyclic topology

The topology of the general networks is depicted in Figure 4.10. Additionally, outgoing arc is added to every node in $A \cup P$. It is linked to a random node in the next stage in order to increase the complexity further while maintaining acyclicness. The approach from Sections 4.5.1-4.5.3 for setting holding costs could not be adopted for general topologies due to its formal complexity. Holding costs



(a) Success ratios



(b) Runtime/evaluations

Figure 4.9.: Success ratios and runtime for convergent networks with 105 stock-points

are 1 for all nodes in A. Holding costs of a node i in P and E are set to

$$h_i = U[1;2] \sum_{j \in v(i)} h_j, \tag{4.27}$$

reflecting a value-added cost structure. Results from the experiments are plotted in Figure 4.11. As foreshadowed by the results for convergent networks, the (1+1)-EA is not able to solve the instances reliably. Interestingly, both the sGA and the EDA are highly reliable when solving instances with 25 stock-points, while a consideration of runtimes is in favor of the EDA. Due to limited computational resources available, we could not solve larger instances of this type.

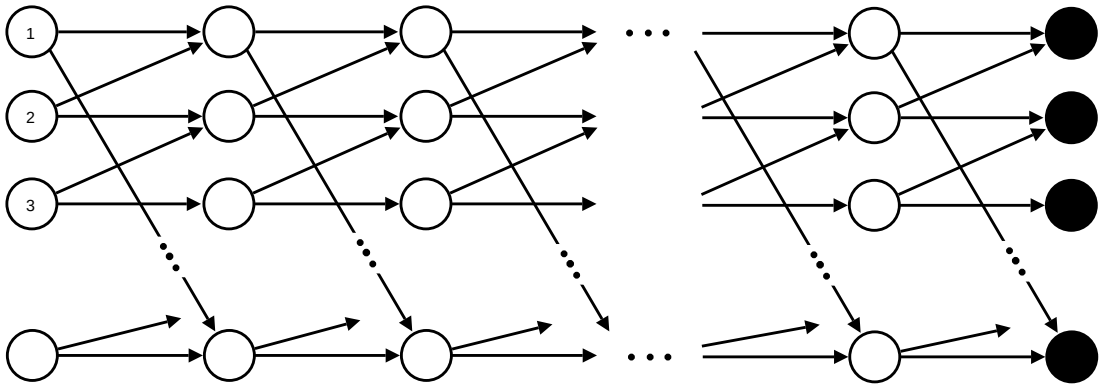
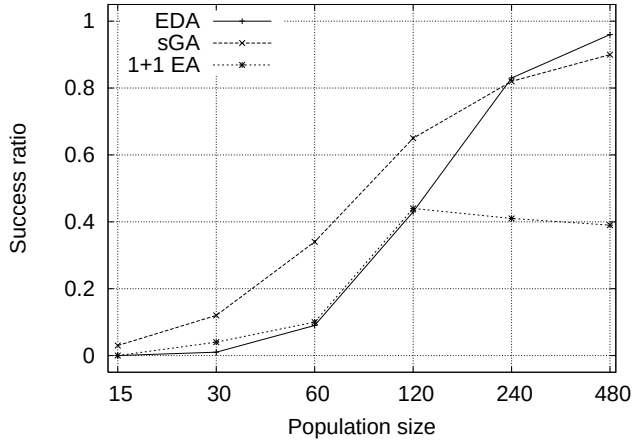
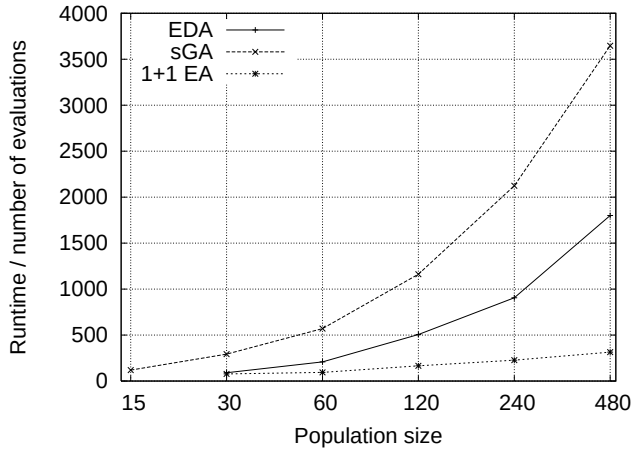


Figure 4.10.: Topology of general networks



(a) Success ratios



(b) Runtime/evaluations

Figure 4.11.: Success ratios and runtime for general networks with 24 stock-points

4.6. Summary and conclusion

We investigated the reliability and runtime of the simple genetic algorithm, the (1+1)-EA and the Bayesian optimization algorithm on safety stock allocation problems. Recall that these algorithms do not exploit more problem specific knowledge than the total expected holding costs caused by a safety stock allocation which can readily be derived on basis of a well-known extreme point property. Nonetheless algorithm-topology matchings exist where the EA solve safety stock allocation problems to global optimality with a reliability of 80% or higher.

For serial topologies, the (1+1)-EA is surprisingly reliable and fast. The Bayesian optimization algorithm can be suggested for divergent, convergent and general acyclic structures. These algorithm-topology matchings proved to be reliable and efficient in our experiments. At the same time, their implementation required minimal effort.

The instances that are considered in this chapter are small compared to instances used, e.g., in Humair and Willems (2006). Specialized heuristics are required to solve large instances. One approach is to hybridize the BBO algorithms used in this section with local search algorithms like Tabu Search (Glover and Laguna (1997)) or Simulated Annealing (Kirkpatrick et al. (1983)). Minner (2000) presents results using local search to solve safety stock allocation problems that can act as starting points.

5. Decomposition of single- and multi-product lot-sizing problems and scalability of EDA

5.1. Introduction

The main contribution of this chapter is twofold. *First*, similar to Chapter 3 the practical relevance of decomposability assumptions is demonstrated. Two lot-sizing problems are studied that are fundamental in inventory management and of practical relevance. These are the single-product lot-sizing problem (Wagner and Whitin (1958)) and the dynamic joint replenishment problem (JRP, Silver (1979)). The single-product lot-sizing problem considers the placement of replenishment orders for a single product over time and is polynomial-time solvable, see Wagelmans and van Hoesel (1992). The dynamic joint replenishment problem extends the single-product problem to a multi-product case where ordering costs are linked between the products. The JRP is NP-complete (Arkin et al. (1989)). In this chapter, it is shown that both problems are decomposable and that their fitness functions can be formulated as additively decomposable functions. It is assessed how Boltzmann-type search distributions factorize into marginal distributions for these decompositions and it is shown that the factorizations of the Boltzmann distribution are polynomially bounded for both problems.

Second, the problems are solved with the BOA, a state-of-the-art EDA in an experimental scalability analysis. The total number of evaluations required to reliably solve the problems to optimality grows with a low-order polynomial depending on the problem size. The results confirm existing scalability theory for decomposable problems and show the potential of EDA in lot-sizing.

This chapter is structured as follows. In Section 5.2.1, a brief introduction to lot-sizing problems is given. The single-product lot-sizing problem is presented in Section 5.2.2, the dynamic joint replenishment problem in Section 5.2.3. In Section 5.3, we propose decompositions for the single- and the multi-product lot-sizing problem. In Section 5.4, experimental scalability analysis for both problems are conducted. The chapter ends with concluding remarks. It has been published as Grahl et al. (2008).

5.2. Lot-sizing

5.2.1. Introduction

Almost any organization has to cope with inventories to exploit economies of scale, guarantee a certain service level or to decouple processes to name just a few reasons. A central problem of managing inventories is matching replenishment ordering decisions with customer demand over time. Lot-sizing problems are solved to balance cost trade-offs that arise from placing replenishment orders in an inventory system at different points in time.

Assume that customer demand for a product is known over time, e.g., for the next 12 weeks. A company can place replenishment orders for the product at various discrete time points, e.g., every Monday. Received but not used goods can be stored as inventories. Customer demand has to be satisfied completely from the available sources.

The total costs that are caused by placing replenishment orders consist of (1) fixed ordering costs and (2) inventory holding costs. Fixed ordering costs arise for each order that is placed (e.g. shipping costs). They are independent from the amount of goods that is actually ordered. Inventory holding costs arise, if goods are put on stock and are carried as inventory. The sum of the fixed ordering costs *grows* with the number of orders that are placed. The sum of the inventory holding costs *decreases* with the number of orders that are placed. This trade-off has to be balanced in order to minimize the total costs that is caused by placing replenishment orders.

5.2.2. Single-product lot-sizing

The standard single-product dynamic, discrete time lot-sizing problem (also known as the Wagner-Whitin problem) was introduced in Wagner and Whitin (1958). This fundamental problem addresses the placement of replenishment orders for a single product over time such that the sum of fixed ordering costs and inventory holding costs is minimized. The single-product lot-sizing problem is the starting point for several extensions into various directions that address issues arising in practice.

In the single-product lot-sizing problem, time is discredited into T time-points. The t-th period denotes the time interval between time-point t and time-point $t+1$. Orders of amount $q_t \geqslant 0$ are placed at each time point $t \in \{1, 2, \ldots, T\}$. This means that orders are placed at the beginning of period t. An order of $q_t > 0$ causes a fixed costs of c that is independent of the amount that is actually ordered. The ordered goods are immediately available. They can be stored in a warehouse with unlimited capacity. The inventory level of the warehouse at the end of period

t is denoted by y_t. We assume that initial inventory y_0 is zero. Holding a single item of inventory from one time-point to the next causes a costs of h. After an order q_t has been placed, demand occurs of size z_t. Without loss of generality we assume throughout the chapter that $z_1 > 0$ in the single-product lot-sizing problem because initial periods with zero demand can be neglected. Demand z_t has to be satisfied completely from the order q_t or from stock (thereby reducing y_t).

The objective is to place the orders q_t such that the sum of fixed ordering costs and inventory holding costs over the planning horizon is minimized. Therefore, the following two questions have to be answered:

1. For which time-points $t \in \{1, 2, \ldots, T\}$ should $q_t > 0$?
2. If $q_t > 0$, how large should q_t be?

Using a mixed integer linear program, the single-product lot-sizing problem can be formulated as follows.

$$\begin{aligned}
\min \quad & f = \sum_{t=1}^{T} (c \cdot x_t + h \cdot y_t) && && (5.1)\\
& y_t = y_{t-1} + q_t - z_t && \forall t = 1, \ldots, T && (5.2)\\
& y_0 = 0 && && (5.3)\\
& q_t \leqslant x_t \cdot \sum_{i=t}^{T} z_i && \forall t = 1, \ldots, T && (5.4)\\
& x_t \in \{0, 1\} && \forall t = 1, \ldots, T &&\\
& y_t, q_t \geqslant 0 && \forall t = 1, \ldots, T &&
\end{aligned}$$

In this formulation, binary variables x_t are introduced to indicate whether an order is placed at time-point t. If $x_t = 1$, then an amount of $q_t > 0$ is ordered. If $x_t = 0$, then no order is given at time-point t. In that case, inequality (5.4) restricts the order amount q_t to zero. The inventory balance equation (5.2) states that the inventory level at the end of period t equals the inventory level at the end of period $t-1$ plus items that are ordered in period t (q_t) minus demand that occurs in period t (z_t). Equality (5.3) states the initial condition that starting inventory is zero.

The single-product lot-sizing problem can be solved in polynomial time. Wagner and Whitin (1958) prove a zero inventory property which reduces the set of potential optima to solutions where batches of consecutive demand are ordered. They proposed a dynamic programming formulation that has a complexity of $O(T^2)$. More efficient algorithms with $O(T \log T)$ complexity have been proposed

by Federgruen and Tzur (1991) and Wagelmans and van Hoesel (1992). Several heuristics for the problem have been proposed, overviews are available, e.g., in Wemmerlöv (1982) , DeBodt et al. (1984) and Jans and Degraeve (2007).

5.2.3. The dynamic joint replenishment problem

Lot-sizing problems become more complex in a multi-product context with dependencies between different products. Examples in practice are products that are manufactured on the same machines (multi-item lot-sizing and scheduling problems) or products that share a common warehouse. In the dynamic joint replenishment problem, the ordering costs depend on the mix of multiple products that are replenished jointly.

The JRP considers $K > 1$ products. For each product $k = 1, \ldots, K$, a single product lot-sizing problem (see Section 5.2.2) has to be solved. Fixed ordering costs c^k (also referred to as minor setup costs), inventory holding costs h^k, and customer demand z_t^k are product-specific. We assume for the JRP throughout the chapter that initial periods where all products have zero demand are neglected. It follows that at least a single product has positive demand in the first period. The products are linked as follows. If at least one product is replenished at time-point t, then an additional order cost of c^o arises (also referred to as major setup costs), independent of the number of products that are actually replenished at time-point t. If no product is replenished at time point t, c^o does not arise at time point t. Note that if several products are ordered at time point t, product specific fixed ordering costs c^k arise for each of these products but c^o arises once. One example in practice is that several products are jointly shipped in a single truck, causing transportation costs of c^o.

If $c^o = 0$, the JRP decomposes into K single-product lot-sizing problems that can be solved independently. If $c^k = 0 \; \forall \; k = 1, \ldots, K$, the JRP can be transferred into one single-product lot-sizing problem by using c^0 as setup costs and summing up demands and holding costs.

The dynamic joint replenishment problem can be formulated as a mixed-integer linear program. The notation used follows the notation that has been introduced in Section 5.2.2. Decision variables and problem parameters are assigned a product index k. The fitness function includes all ordering and inventory holding costs.

$$\min \quad f = \sum_{k=1}^{K} \sum_{t=1}^{T} \left(c^k \cdot x_t^k + h^k \cdot y_t^k \right) + \sum_{t=1}^{T} c^o \cdot w_t \tag{5.5}$$

$$y_t^k = y_{t-1}^k + q_t^k - z_t^k \qquad \forall t = 1, \ldots, T; k = 1, \ldots, K$$

$$y_0^k = 0 \qquad \forall k = 1, \ldots, K$$

$$q_t^k \leqslant x_t^k \cdot \sum_{i=t}^{T} z_i^k \qquad \forall t = 1, \ldots, T; k = 1, \ldots, K$$

$$x_t^k \leqslant w_t \qquad \forall t = 1, \ldots, T; k = 1, \ldots, K \tag{5.6}$$

$$y_t^k, q_t^k \geqslant 0 \qquad \forall t = 1, \ldots, T; k = 1, \ldots, K \tag{5.7}$$

$$x_t^k \in \{0, 1\} \qquad \forall t = 1, \ldots, T; k = 1, \ldots, K \tag{5.8}$$

This formulation is a direct extension of the single-product lot-sizing mixed-integer linear program (see Section 5.2.2) for K products. If at least one product is replenished in period t, an indicator variable w_t is forced to 1 and ordering costs c^o arise in the fitness function (5.5). This coupling of the ordering costs is modeled in inequality (5.6).

Algorithms that determine an optimal solution using dynamic programming approaches were developed by Silver (1979) and Kao (1979). A branch-and-bound method was proposed by Erenguc (1988). A dual-based method was developed by Robinson and Gao (1996), heuristics were proposed in Joneja (1990) and Robinson et al. (2007). Efficient integer programming formulations are available in Boctor et al. (2004) and Narayanan and Robinson (2006).

5.3. Decomposition of lot-sizing problems

A major result of EDA theory is that if the factorization of the Boltzmann distribution for a given problem is polynomially bounded, new solutions can efficiently be generated and an EDA can theoretically solve the decomposable problem with a polynomial number of fitness evaluations (Mühlenbein and Mahnig (1999)). During the last years, this potential has been turned into scalable optimizers that outperform standard GA on a wide range of problems.

5.3.1. Single-product case

In this section, we reformulate the fitness function (5.1) as an additively decomposable function of type (2.3). Then, we show that the Factorization Theorem holds for the decomposition and that the size of the marginal distributions is bounded with the consequence that a state-of-the-art EDA should be able to solve the problem in polynomial time.

The proposed decomposition is constructed by exploiting the zero inventory property (see Wagner and Whitin (1958)). This property states that in an optimal solution for the single-product lot-sizing problem, replenishment orders $q_t > 0$ are placed at time t if and only if $y_{t-1} = 0$. From this it follows that order quantities q_t in optimal solutions are batches of aggregate consecutive future demands. In the previous formulation of the problem (see Section 5.2.2), a company has to decide when replenishment orders should be placed *and* how much should be ordered. If we exploit the zero inventory property, it is *only* necessary to decide when the orders have to be placed and not how much should be ordered. The order amounts are completely derived from the times when orders are placed. If an order q_t is placed at time t and the next order u is placed at time $t < u < T$, then $q_t = \sum_{i=t}^{u-1} z_i$. Consequently the last order amount is $q_u = \sum_{i=u}^{T} z_i$.

Exploiting the zero-inventory property allows us to represent a solution on a binary string. The ordering decision variables x_t are binary (either an order is placed at time t or not) and their values are aligned on the string. The ordering decision x_t is represented by the t-th bit on a string of length T.

Note that for lot-sizing problems considered in this chapter an order has to be placed in period one as initial inventory is zero and $z_1 > 0$. This means that solutions where $x_1 = 1$ are feasible and solutions where $x_1 = 0$ are infeasible. For all following reformulations and analysis we assume feasibility of the solutions.

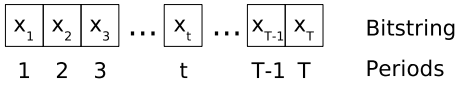


Figure 5.1.: Coding for the single-product lot-sizing problem

The total costs caused by m replenishment orders over T periods of time can be calculated by adding up m costs values that cover the costs that occurs in m lots over disjoint and aligned sets of periods. Thus, the single-product lot-sizing problem is decomposed along batches of aggregate consecutive future demands.

We reformulate (5.1) to an additively decomposable function of type (2.3) as follows:

$$
\begin{aligned}
\min\ f &= \sum_{i=1}^{m} f_i(x_{s_i}) && (5.9)\\
&\text{with} \\
p &= \{p_1, p_2, \ldots, p_m\} \\
p_i &\in \{1, 2, \ldots, T\} && i = 1, 2, \ldots, m \\
p_1 &= 1 \\
s_i &= \bigcup_{j=p_i}^{p_{(i+1)}-1} j. && i = 1, 2, \ldots, m \\
f(x_{s_i}) &= c + h \cdot \left[\sum_{j=0}^{|x_{s_i}|-1} j \cdot z_{(p_i+j)} \right]
\end{aligned}
$$

Fitness function (5.9) is additively defined over m sets of the ordering decision variables x_{s_i}. m, denotes the number of $1 \leqslant m \leqslant T$ orders that are placed. The x_{s_i} are disjoint subsets of all ordering decision variables and x_{s_i} consist of all variables that are associated with the order point p_i. Order points $p_i, i = 1, 2, \ldots, m$ are time points t where $x_t = 1$. The set p contains all order points.

Example 1 *Let the number of periods T be 12. Orders are placed at the beginning of periods 1,5,6, and 10. In this case, $m = 4$, $p = \{1, 5, 6, 10\}$, $p_1 = 1$, $p_2 = 5$, $p_3 = 6$ and $p_4 = 10$. The sets s_i have the following structure: $s_1 = \{1, 2, 3, 4\}$, $s_2 = \{5\}$, $s_3 = \{6, 7, 8, 9\}$, and $s_4 = \{10, 11, 12\}$. The decision variables x are grouped into sets of $x_{s_1} = \{x_1, x_2, x_3, x_4\}$, $x_{s_2} = \{x_5\}$, $x_{s_3} = \{x_6, x_7, x_8, x_9\}$, and $x_{s_4} = \{x_{10}, x_{11}, x_{12}\}$. These groups can be denoted as building blocks of the problem instance.*

The example illustrates that the building blocks of single-product lot-sizing problem instances have a well-defined structure. They consist of a leading 1 that denotes the setup decision plus subsequent 0s, if any. An illustration of the BB structure is given in Figure 5.2. The BBs of a problem instance need not be equally sized. But the average size of the BBs will increase with the setup costs c and decrease with h. The more expensive an order is, the more is ordered in a single batch to avoid frequent ordering and the more demand of future periods is covered by a single order.

We now show that the Factorization Theorem holds for this decomposition and that the factorization of the Boltzmann distribution is polynomially bounded.

First, we show that $b_i \neq \emptyset$ for all $i = 1, 2, \ldots, m$.

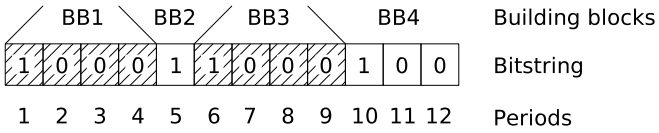


Figure 5.2.: Building blocks of a single-product lot-sizing instance

Obviously, all $s_i \neq \emptyset$, because the s_i always include at least one index p_i. Since $s_i \cap s_j = \emptyset$ for all $i \neq j$, it follows that

$$c_i = s_i \cap d_{i-1} = s_i \cap \bigcup_{j=1}^{i-1} s_j = \emptyset.$$

Thus, $b_i = s_i \setminus d_{i-1} = s_i$. Since $s_i \neq \emptyset$, it follows that $b_i \neq \emptyset \ \forall i = 1, 2, \ldots, m$. The RIP is fulfilled because the sub-problems do not overlap, the problem is separable.

Additionally, it is desired that the factorization is polynomially bounded. This means that the size of the marginal distributions is bounded by a constant independent of T. The size of a marginal distribution relates directly to the number of periods $|s_i|$ whose demand a replenishment order placed in p_i covers. It is reasonable to assume that the ordering costs c and the inventory holding costs h are bounded and positive. Assume now that $T \to \infty$. Under these assumptions, the number of periods that any replenishment order covers is bounded. Any values chosen for c and h will make it beneficial to order more than once if $T \to \infty$. The size of any s_i is bounded. b_i is bounded since $b_i = s_i$ and c_i is bounded because $c_i = \emptyset$. The factorization of the Boltzmann distribution is polynomially bounded. The planning horizon theorem from Wagner and Whitin (1958) yields similar results by stating independence of partial solutions spanning periods t to $t + H$, if the optimal solution for periods t to $t + H + 1$ includes an additional order in period $t + H + 1$ and the optimal order amount in period t remains unchanged.

Chapter 3 discussed that the numbering of warehouses in a warehouse location problem causes performance differences when the WLP is solved with sGA. The numbering effect is caused by encoding bits that belong to a BB loosely on the bitstring. The codings that are presented in Figures 5.1 and 5.3 map the timely sequence of ordering decisions directly onto the genotype. The bits that belong to a BB are coded next to each other. For the single-product lot-sizing problems that are studied in this chapter the numbering effect is thus assumed to be neglectable.

5.3.2. Multi-product case

In this section, we will reformulate (5.5) as an additively decomposable function of type (2.3). Then we show that the Factorization Theorem holds for this decomposition and that the factorization of the Boltzmann distribution is polynomially bounded.

The zero-inventory property also holds for the dynamic joint replenishment problem (Veinott (1969)). This allows us to extend the additively decomposable reformulation of the single-product lot-sizing problem (see Section 5.3.1) to the multi-product case.

The reformulation introduces an artificial product with index $k = 0$ that is used to model the coupling of ordering costs. Inventory holding costs for the artificial product are zero. If at least one real product is replenished at time point t, then the artificial product is replenished as well at time point t. This causes fixed ordering costs of c^o that are independent of the number of products that are actually replenished in period t. If no real product is replenished at time-point t, then the artificial product is not replenished at time-point t.

Solutions of the JRP are represented on a binary string as follows. Like in the single-product case we do not need to encode the amounts that are ordered. Due to the zero-inventory property, we only need to encode binary ordering timing decisions. A single product $k = 0, 1, \ldots, K$ is exactly encoded as described in 5.3.1. The setup decisions for all products are aligned on the binary string. This means, that if 3 products are given and $T = 12$, then $4 \cdot 12 = 48$ bits are needed to encode a single solution. The first T bits are used to represent the order decision of the artificial product. Bits $T + 1$ to bit $2 \cdot T$ represent ordering decisions of the first product, second, and so forth. In general, the ordering decision for period t of product k is represented by bit $T \cdot k + t$. The coding is illustrated in Figure 5.3.

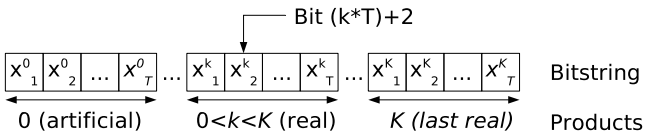


Figure 5.3.: Coding for the dynamic joint replenishment problem

We reformulate function (5.5) as an additively decomposable function of type (2.3) as follows:

$$\min\ f(x) = \sum_{k=0}^{K} \sum_{i=1}^{m^k} f_i^k \left(x_{s_i^k} \right) \tag{5.10}$$

with

$$\begin{aligned}
& c^0 = c^o; \quad h^0 = 0 \\
& p^k = \{p_1^k, p_2^k, \ldots, p_{m^k}^k\} && \forall k = 0, 1, \ldots, K \\
& p_i^k \in \{1, 2, \ldots, T\} && \forall k = 0, 1, \ldots, K; i = 1, 2, \ldots, m^k \\
& p_1^k = 1 && \forall k = 1, 2, \ldots, K \\
& p^k \subseteq p^0 && \forall k = 1, \ldots, K \qquad (5.11) \\
& s_i^k = \bigcup_{j=p_i^k}^{p_{(i+1)}^k - 1} [\, j + (k \cdot T)] && \forall k = 0, 1, \ldots, K; i = 1, 2, \ldots, m^k.
\end{aligned}$$

$$f_i^k(x_{s_i^k}) = c^k + h^k \cdot \left[\sum_{j=0}^{|x_{s_i^k}|-1} j \cdot z_{(p_i^k + j)}^k \right]$$

Fitness function (5.10) is additively defined over all products $k = 0, 1, \ldots, K$ and all associated sets of decision variables $x_{s_i^k}$ for each of the products. (5.11) links the ordering decisions of the real product with the ordering decisions of the artificial product.

Example 2 *Let the number of periods T be 12 and the number of real products K be 3. The first product is replenished in periods 1,5, and 6. The second product is replenished in periods 1, 6, and 11 and the third product is replenished in periods 1 and 3. The artificial product is therefore replenished in periods 1,3,5,6, and 11. For this setting, $p^1 = \{1, 5, 6\}$, $p^2 = \{1, 6, 11\}$, $p^3 = \{1, 3\}$ and $p^0 = \{1, 3, 5, 6, 11\}$. The set $s_1^0 = \{1, 2\}$, $s_2^0 = \{3, 4\}$, $s_3^0 = \{5\}$, $s_4^0 = \{6, 7, 8, 9, 10\}$, $s_5^0 = \{11, 12\}$, $s_1^1 = \{13, 14, 15, 16\}$, $s_2^1 = \{17\}$, $s_3^1 = \{18, 19, 20, 21, 22, 23, 24\}$, $s_1^2 = \{25, 26, 27, 28, 29\}$, $s_2^2 = \{30, 31, 32, 33, 34\}$, $s_3^2 = \{35, 36\}$, $s_1^3 = \{37, 38\}$, $s_2^3 = \{39, 40, 41, 42, 43, 44, 45, 46, 47, 48\}$.*

We have shown in Section 5.3.1 that the Factorization Theorem holds for a single product. We transfer this result to the dynamic joint replenishment problem. The above reformulation of the JRP separates the products from each other. The index sets s_i^k do not overlap between the products. This means that for each product that is modeled in the JRP, the single-product results apply. Because the products of the JRP are separated, the Factorization Theorem also holds for the JRP as a whole.

We have shown in Section 5.3.1 that the Boltzmann distribution is polynomially bounded for a single product. In our JRP formulation we separate the products from each other and no additional complexity is introduced for all feasible solutions. Thus, this result is valid individually for every "real" product $k > 0$. Note, that the artificial product is special because it has zero-inventory holding costs. The size of the inventory holding costs does not affect the analysis of the decomposition. Therefore, the results for a single real product apply for the artificial product as well.

An instance of the JRP is made up by a combination of a bounded number of real products and one single artificial product. If we let $T \to \infty$, the size of no set s_i^k can tend to infinity. All sets s_i^k are bounded. All sets b_i^k are bounded because $b_i^k = s_i^k$. The sets c_i^k are bounded because $c_i^k = \emptyset$. The factorization of the Boltzmann distribution is polynomially bounded for the JRP as a whole.

Renumbering effects that result from changing the allocation of bits to decision variables might occur when different sorting of products are used. For one product k, the mapping the timely sequence provides a naturally tightly linked encoding.

5.4. Experimental results

We perform experiments on the single-product lot-sizing problem and the dynamic joint replenishment problem. The analysis of the previous chapters indicates that both problems are decomposable. State-of-the-art EDA solve decomposable problems in polynomial time. We conduct an experimental scalability analysis to assess the degree of the polynom and how the constant factor depends on the problem instance.

The Hierarchical Bayesian Optimization Algorithm (hBOA) (Pelikan et al. (1999); Pelikan (2002)) is used in all experiments because it is a state-of-the-art EDA. It has been used for solving complicated problems from computer science and physics previously, see Pelikan and Goldberg (2003) and Chapter 2.2.4. hBOA performs selection by restricted tournament replacement (RTR). In RTR, a specified percentage of the population is iteratively replaced by the best individual out of a randomly chosen subset of individuals of size w (called the window size). We chose to replace 50% of the population. After some initial testing, the window size for RTR was set to 4, independent of the size of the problem l.

5.4.1. Single-product dynamic lot-sizing

We implemented fitness function (5.9) for hBOA. Solutions were encoded on the binary string as explained in Section 5.3.1. In every generation of the algorithm,

the feasibility of the population was maintained by setting the first bit of a solution to 1.

In order to assess whether the scalability results for the lot-sizing problem are in accordance with scalability theory, we propose two test problems with constant demand and *constant* BB-size in Table 5.1. In the optimal solution of the first test problem, BBs are of size 2. In the optimal solution of the second test problem, BBs are of size 6.

Table 5.1.: Test problems with constant BB-sizes for the single-product lot-sizing problem

BB-size	Optimal costs	c	h	$z_t \; \forall t$
2	$T/6 \cdot 5400$	15000	8	100
6	$T/6 \cdot 27000$	1500	8	100

We propose a second test problem with seasonal demand in Table 5.2 where BBs of the optimal solution have *varying* sizes from 2-6.

Table 5.2.: Test problem with varying BB-sizes for the single-product lot-sizing problem

BB-size	Optimal costs	c	h	z_t	
$2-6$	$T = 6 : 3680$	1000	2	$z =$	(100,140,180,140,120,110,
	$T = 12 : 6100$				80,50,30,50,80,90,
	$T = 18 : 9600$				100,140,180,140,120,110,
	$T = 24 : 12080$				80,50,30,50,80,90,
	$T = 30 : 15680$				100,140,180,140,120,110,
	$T = 36 : 18160$				80,50,30,50,80,90,
	$T = 42 : 21680$				100,140,180,140,120,110,
	$T = 48 : 24160$				80,50,30,50,80,90,
	$T = 54 : 27760$				100,140,180,140,120,110,
	$T = 60 : 30240$				80,50,30,50,80,90)

For all problems, we varied the problem size T from 6 to 60 in steps of size 6, resulting in 10 problem sizes per problem. For each problem size, the optimal solution was computed with the solver XPress-MP as listed in the tables. Then, we derived the minimal population size that hBOA required to solve the problem to optimality using a bisection method. An instance was assumed to be solved to optimality, if in at least 27 out of 30 independent consecutive runs of hBOA, the entire population converged towards the optimal solution. For the minimally required population size, the number of fitness evaluations was averaged over the number of successful runs.

Figure 5.4 illustrates how the average number of fitness evaluations depends on the problem size l. Additionally, the average number of evaluations has been approximated by a function of the form $O(l^r)$, where r was set such that experimental results were fitted accurately using regression. More stable experimental results for larger problems where emphasized, smaller instances were neglected due to high volatility of the results. Straight lines in the plot indicate polynomial scalability.

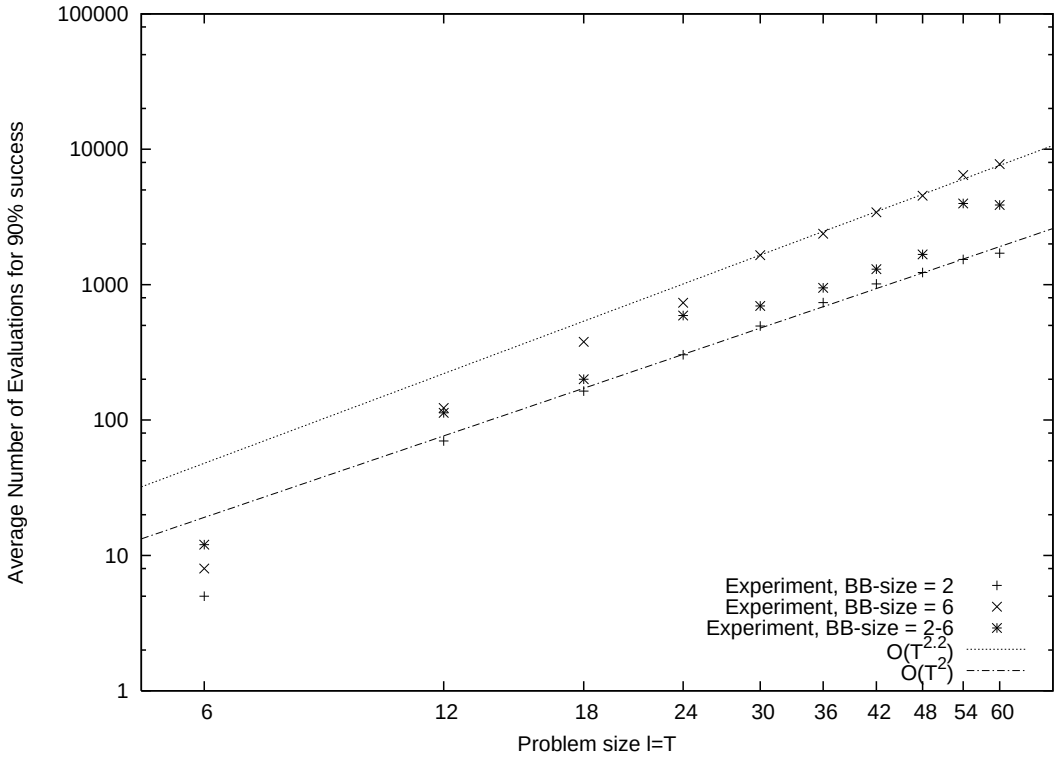


Figure 5.4.: Scalability results for the single-product lot-sizing problem

As can be seen in Figure 5.4, the average number of fitness evaluations grows with a low-order polynomial depending on the problem size $l = T$. For a BB-size of 2, r resulted in $r = 2.0$, for a BB-size of 6, r resulted in $r = 2.2$. hBOA scales within these bounds, if the BB-size is in between $2 \leq \text{BB-size} \leq 6$.

Scalability theory for hBOA on non-hierarchical problems predicts that the number of fitness evaluations needed to reliably solve decomposable problems of bounded order grows with a low-order polynomial depending on the problem size with respect to the problem size l (Pelikan (2002)). Our estimate of r for a BB-size of 6 lies slightly above quadratic scalability. Still, hBOA succeeds in solving single-product lot-sizing problems within low-order polynomial time.

The size of the constant factor of the polynom that approximates the experimental results grows with the size of the BBs. This means that hBOA needs more time

to solve instances where large batches are ordered, compared to instances where smaller batches are ordered. If batches are not sized identically, the constant factor grows with the size of the largest batch ordered in the optimal solution.This is in accordance with scalability theory for EDA where the size of a BB drives the runtime of EDA, see Pelikan et al. (2003).

5.4.2. The dynamic joint replenishment problem

We implemented fitness function (5.10) in hBOA. Solutions were represented as described in Section 5.3.2. Feasibility of all solutions was maintained as follows in each generation. We assume positive demand. All bits $T*k \; \forall \; k = 0, 1, 2, \ldots, K-1$ were set to 1, if they were 0. Additionally, whenever $\sum_{k=1}^{K} x_{(k*T)+j} = 0$ for any $j = 0, 1, 2, \ldots, T-1$, then x_j was set to 0. This means that if none of the real product is ordered at time-point j, the artificial product is not ordered as well.

We conduct scalability analysis for $K = 2$ and $K = 6$ products. For each case, we propose in Table 5.3 two test problems with constant demand and constant BB-size, resulting in 4 problems in total. For both $K = 2$ and $K = 6$, a problem with BB-size of 2 and a problem with BB-size of 6 is designed. Just like in the single-product case, we expect that a problem instance with seasonal demand and varying BB-size between 2 and 6 would scale up inside the bounds of these problems.

Table 5.3.: Test problems with constant BB-sizes for the JRP

	BB-size	Optimal costs	$z_t^k \; \forall \; t, k$	$h^k \; \forall \; k$	c^o	$c^k \; \forall \; k$
2	2	$\frac{l}{18} \cdot 11400$	100	8	2000	100
Products	6	$\frac{l}{18} \cdot 3260$	10	5	1560	100
6	2	$\frac{l}{42} \cdot 2100$	10	5	100	50
Products	6	$\frac{l}{42} \cdot 9100$	10	5	1600	500

By varying the number of time-points T between 6 and 60 with a step size of 6, we obtained 10 problem instances for the 2 products case. The scalability analysis for two products spans problem sizes from 18 to 180 bits. For the 6 products case, T was varied from 6 to 30 in steps of 6, yielding 5 problem instances. This was necessary due to limited computational resources available for doing the scalability analysis. The scalability analysis for $K = 6$ products spans problem sizes from 42 to 210 bits.

For each instance, we obtained the optimal costs using the solver XPRESS MP as listed in Table 5.3. We derived the minimal population size that hBOA required to solve the problem to optimality using a bisection method. A problem instance was solved to optimality, if in at least 27 out of 30 consecutive and independent runs of hBOA, the optimal solution was found. For the minimal required population

size, the average number of fitness evaluations was averaged over all successful runs.

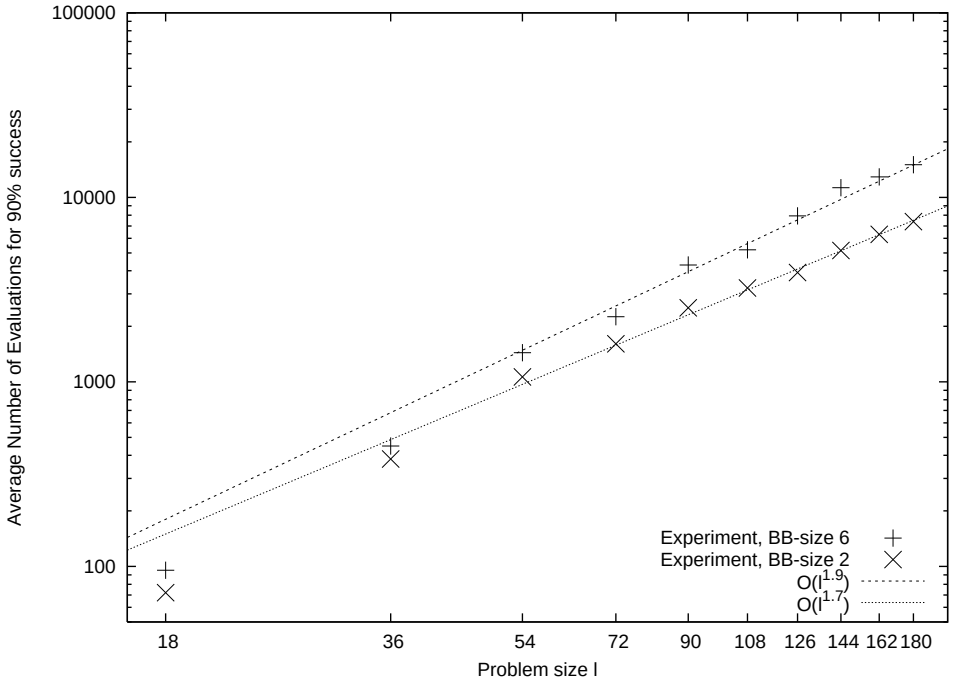
Figure 5.5 illustrates how the average number of fitness evaluations depends on the size of the problem for $K = 2$ and $K = 6$ products. In addition, the average number of evaluations has been approximated by a function of the form $O(l^r)$, where r was set such that experimental results were fitted accurately, again emphasizing larger problem sizes. Both plots have a log-log scale. Straight lines in the plots indicate polynomial scalability.

For the case that the number of products $K = 2$, r has been set to $r = 1.9$ (BB-size of 6) and $r = 1.7$ (BB-size of 2). For the 6 products case, $r = 2.5$ (BB-size of 6) and $r = 2.4$ (BB-size of 2). Thus, hBOA succeeds in solving decomposable instances of the JRP in low-order polynomial time. For the two-products case, the constant factor grows with the size of the BBs. Like in the single-product case, more time is needed to solve instances of the JRP where larger batches are ordered optimally, compared to instances of the JRP where smaller batches are ordered.

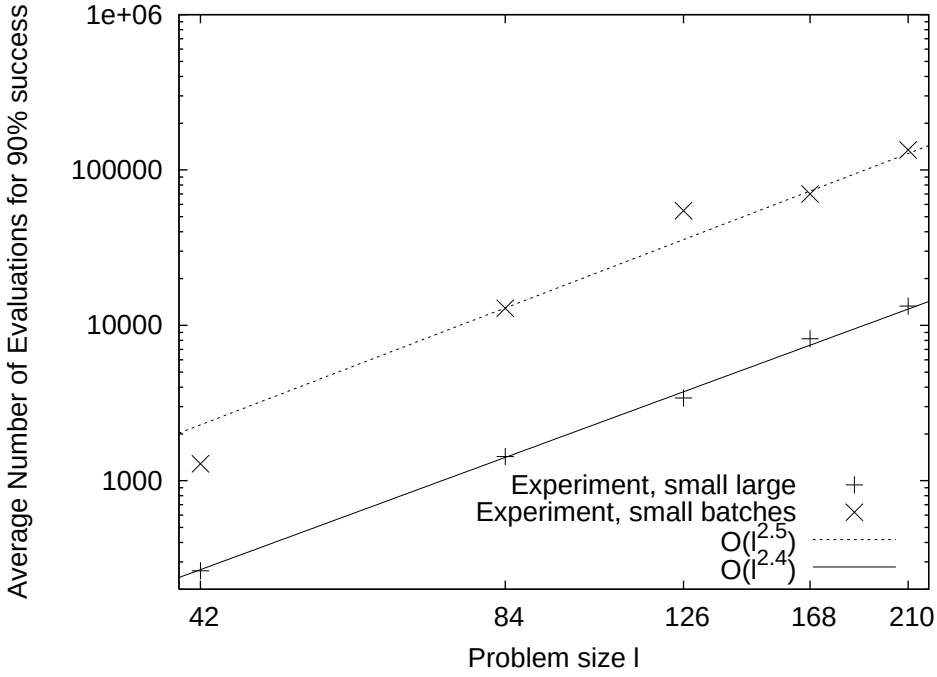
The scalability results from the 6-products case illustrated in Figure 5.5b) yield counter-intuitive results. hBOA scales polynomially for small and large BB-sizes, but obviously the problem set with a BB-size of 2 is harder to solve and requires more fitness evaluations than the corresponding instance with a BB-size of 6. This result can be explained with the population sizing model from Harik et al. (1999) which states that the required population size scales inversely proportional to the signal-to-noise ratio defined as σ_{bb}/d. Noise σ_{bb} denotes the standard deviation of fitness values from all solutions that include the best BB and indicates the amount of fitness variability in the instance. The signal d denotes the difference between the mean fitness $\bar{f}_1$ of solutions that contain the best BB and the mean fitness $\bar{f}_2$ of solutions that contain the second-best BB. The signal-to-noise ratio for the JRP instance with 6 products and 6 periods ($l = 42$) is presented in Table 5.4. Note that, for a BB-size of 2, the average costs of solutions that contain the second-best BB is lower than that of solutions that contain the best BB. This deception is not present in the case of BB-size 6, rendering the instance with smaller BB-size harder to solve than the instance with larger BBs.

Table 5.4.: Signal-to-noise ratio for JRP with six products

	$\bar{f}_1$	$\bar{f}_2$	d	σ_{bb}	Signal-to-noise ratio
BB-size 6	20107.81	20257.80	-150.01	1717.19	-0.09
BB-size 2	2864.06	2823.44	40.62	309.92	0.13



(a) 2 Products



(b) 6 Products

Figure 5.5.: Scalability results for hBOA on the joint replenishment problem

5.5. Summary and conclusion

Decomposability of fitness functions is well-understood and frequently assumed in theoretical GA and EDA literature. State-of-the-art EDA reliably solve decomposable problems in a low-order polynomial number of fitness evaluations depending on the problem size. This success makes it a tempting idea to apply EDA to problems of practical relevance. It is essential to bridge the gap between theoretical work that focuses on solving decomposable problems and applied work that focuses on solving problems of practical interest. However, the complexity of the real world makes a direct adaption of theoretical concepts a stiff task and it is rarely known which problems are decomposable.

In this chapter, we demonstrated that decomposability is of practical relevance and valid for certain problems in inventory management. The decomposability of single-product lot-sizing and the dynamic joint replenishment problem was analyzed. The results indicated that these lot-sizing problems are indeed decomposable into sub-problems of bounded complexity. We conducted a scalability analysis that showed that a state-of-the-art EDA can reliably solve the problems with a low-order polynomial number of fitness evaluations depending on the problem size. The results are promising and reveal the potential of EDA applications in inventory management.

Part II.

Analysis and design of continuous EDA

6. Convergence phases

6.1. Introduction

Over the years, the focus of published work on continuous EDA has changed with respect to at least the following issues. In first-stage EDA the correct choice of the *structure* of probabilistic models was regarded as crucial for efficient optimization. This was largely motivated through the lessons that had been learned from the analysis of the dynamics and design of discrete EDA. However, the results obtained in the continuous domain were by far not comparable to that of their discrete counterparts and these algorithms sometimes failed on problems where much simpler algorithms, even hill-climbers, did not fail. Results like those presented in the following chapters let researchers conclude that a sensible adaption of the *parameters* of the model (specifically the variances) boosts the performance of continuous EDA and requires deeper analysis.

Furthermore, while most of the initial work was experimental, researchers have started to model the dynamics of continuous EDA, see González et al. (2002) and Yuan and Gallagher (2006). In addition to being interesting itself, formal analysis provides guidelines for design decisions.

The contributions of the following chapters are in line with these trends. We present a formal study of the hill-climbing behavior of continuous EDA. The EDA is initialized with a mean that is far from the optimal solution and should minimize the sphere function, that is simply the sum of squared values for each considered dimension, to a pre-defined value to reach. All points that have a fitness smaller or equal a given value to reach lie inside an elliptical convex region. This region is called the optimal region. The convergence process is artificially decomposed into three phases, see Figure 6.1. (1) The mean is far from the optimal region and optimal solutions have a negligible chance of being sampled. (2) The mean is close to, but still outside, the optimal region and significant proportions of the sampled candidate solutions are in the optimal region. (3) The mean is positioned on the optimum.

For phase (1) we replace the sphere function by a linear function as a substitute for slope-like regions. In Section 7 we derive closed form expressions of the population statistics of a simple EDA in phase (1). It becomes obvious that continuous EDA that use maximum-likelihood estimates to sample offspring are likely to converge prematurely on slope-like regions of the search space. The reason is that the variance decreases too fast if maximum likelihood estimates

are used to sample offspring. We show in Section 8.1 that once an EDA is in phase (2), a unique, optimal sampling variance exists, that can be obtained in a closed form. The optimal sampling variance maximizes the proportion of solutions that are optimal. For convergence in phase (3) we derive in Section 8.2 a lower bound on the number of generations that is required until the optimal solution is sampled with, e.g., 99.5% chance. Such runtime analysis is available for, e.g., Evolution Strategies in Jägersküpper and Witt (2005) and Jägersküpper (2005), but is still missing for continuous EDA.

These novel results are discussed in Section 8.3 with a special focus on how they influence design guidelines for continuous EDA. It should be noted, that the following chapters present results that are obtained under limiting assumptions such as the use of normal distributions, maximum likelihood estimation of model parameters and an infinite population size. The results from Chapters 6-8 have been published in Grahl et al. (2005) and Grahl et al. (2007a).

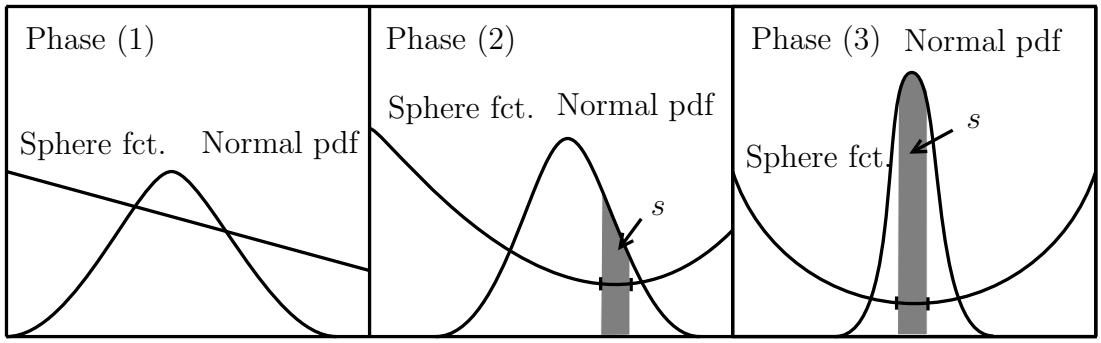


Figure 6.1.: Decomposition of the overall process into three artificial phases. s denotes the success ratio, that is the mass of the normal pdf inside the optimal region.

6.2. Notation and algorithm

Index t denotes the state of a variable in generation t. As an example, a variance in generation t is denoted by σ_t^2. Furthermore, the $(1-\alpha)$-quantile of the chi-square distribution with n degrees of freedom is $\chi^2_{\alpha,n}$. The standard normal density at value x is $\phi(x)$. $\phi_{\mu,\sigma}(x)$ denotes the normal density with mean μ and standard deviation σ at value x. Cumulative standard normal density at value x are denoted by $\Phi(x)$. The quantile function of the standard normal density (the $x \cdot 100\%$ quantile) is $\Phi^{-1}(x)$. Similarly, $\Phi^{-1}_{\mu,\sigma}(x)$ denotes the quantile function of the normal distribution with mean μ and standard deviation σ. The cumulative density of normal distribution with mean μ and standard deviation σ, at value x is $\Phi_{\mu,\sigma}(x)$. Accordingly, $\Phi_{\mu,\sigma^2}(x)$ denotes the cumulative density of a normal

distribution with mean μ and variance σ^2 at value x. τ denotes the percentage of selected solutions in truncation selection. v is a target fitness function value called the value to reach. Different to previous chapters, $\boldsymbol{x} = (x_1, x_2, \ldots, x_l)$ is a single solution.

This chapter analyzes the UMDA$_c$ algorithm with truncation selection. UMDA$_c$ is a simple EDA introduced in Larrañaga et al. (2000a), a description is available in Section 2.3.1. In UMDA$_c$, the essential parameters are μ_i^t and σ_i^t. We are interested in how these parameters change over time. Therefore we make the following assumptions.

We assume an infinite population size. Furthermore, we minimize a l-dimensional function f to by using the UMDA$_c$ algorithm to the vector $\boldsymbol{x}^\star$ that minimizes f.

The sphere function assigns a single, $l-$dimensional solution $\boldsymbol{x} = (x_1, x_2, \ldots, x_l)$ a fitness

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} x_i^2.$$

A value to reach is a real value v that denotes the maximal fitness that a solution may have to be considered optimal. Further, we refer to the estimated variance as the maximum-likelihood variance (ML-variance). A variance that is used to sample offspring from is referred to as a sampling variance. Note that in original UMDA$_c$, the ML-variance equals the sampling variance. If the ML-variance is modified before sampling, e.g., because it is scaled, the sampling variance can be different.

7. UMDA$_c$ on monotonous functions

In this chapter, we center our analysis on convergence behavior of UMDA$_c$ in phase (1). We present an approach to model the behavior on the set of monotonous functions. We focus on monotonous functions in order to model UMDA$_c$'s behavior when it is far from the optimum in unimodal search spaces. In a first step, we analyze the effect of the truncation selection scheme for use with monotonous fitness functions in Section 7.1 and link it to the behavior of UMDA$_c$ in Section 7.2. Consequently, analytical results are derived. First, we show how population statistics change from generation t to generation $t + 1$ (Sections 7.3 and 7.4), then we analyze population statistics in generation t (Section 7.5). Finally, we investigate convergence behavior of UMDA$_c$ in Section 7.6 and discuss the results.

7.1. Monotonous fitness functions and truncation selection

In order to use UMDA$_c$ a fitness function f needs to be specified for evaluating the population. In many work on parameter optimization, highly complex fitness functions are used as benchmark problems. These functions exhibit deceptive structures, multiple optima, and other features that makes optimizing them a stiff test.

We make simplified assumptions on the structure of f. In particular, we focus on the behavior of UDMAc on monotonous functions. This set of functions includes, but is not limited to linear ones. This can be seen as a way to model the structure of the search space far away from the optimum. A nice side effect is reduced complexity of our analytical analysis.

In the following we will introduce monotonous fitness functions. Furthermore, We will show an interesting result that occurs when combining truncation selection with monotonous fitness functions. This result will be of some importance later on.

Let S be a population of individuals. Let $\boldsymbol{x}^j$ and $\boldsymbol{x}^k \in S$ be two distinct individuals of the population and let $g_i : \mathbb{R} \to \mathbb{R}$ be a fitness function defined over the

$i-th$ gene of the individuals (denoted as $\boldsymbol{x}_i^j$ and $\boldsymbol{x}_i^k$). Then,

$$\begin{aligned} &g_i \text{ is increasing if } \boldsymbol{x}_i^j \leq \boldsymbol{x}_i^k \text{ implies that} \\ &\quad g_i(\boldsymbol{x}_i^j) \leq g_i(\boldsymbol{x}_i^k) \; \forall \; \boldsymbol{x}^j, \boldsymbol{x}^k \in S \\ &g_i \text{ is decreasing if } \boldsymbol{x}_i^j \leq \boldsymbol{x}_i^k \text{ implies that} \\ &\quad g_i(\boldsymbol{x}_i^j) \geq g_i(\boldsymbol{x}_i^k) \; \forall \boldsymbol{x}^j, \boldsymbol{x}^k \in S. \end{aligned} \tag{7.1}$$

We consider a fitness landscape monotonous if the fitness function f is either increasing or decreasing. Note that the class of monotonous functions includes, but is not limited to linear functions.

Assume that a population P of individuals is given. We use l different increasing functions $f_{1...l}$ to evaluate this population P. After each evaluation process, we use truncation selection of the best $\tau \cdot 100\%$ of the individuals and call the l sets of selected individuals $M_{1...l}$. It is a simple, yet interesting fact that all sets $M_{1...l}$ have to be identical. Note that the fitness of the selected individuals may of course be different. For our analysis it is more important that the selected individuals are identical. Note that if all fitness functions are decreasing, this fact is true as well.

In the density estimation process of UMDA$_c$, the fitness of the individuals is not considered. Density estimation solely relies on the genotypes of the selected individuals, which is the $\boldsymbol{x}$. As the parameters μ_i^t and σ_i^t are estimated from the $\boldsymbol{x}$, they are identical for all $f_{1...l}$.

This fact simplifies our further analysis. We can now state that the UMDA$_c$ will behave the same for all increasing fitness functions (and for all decreasing functions). Thus, we can base our analysis on the simplest monotonous function that is the linear one. Yet, we know that our results are valid for all monotonous functions.

7.2. UMDA$_c$ for monotonous fitness functions

In the following paragraphs, we model the behavior of UMDA$_c$ with truncation selection on fitness functions of the type

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} g_i(x_i), \tag{7.2}$$

where $g_i(x_i)$ is an increasing function. Note, that the case of decreasing functions does not provide additional insight. Thus, we focus our analysis on increasing functions.

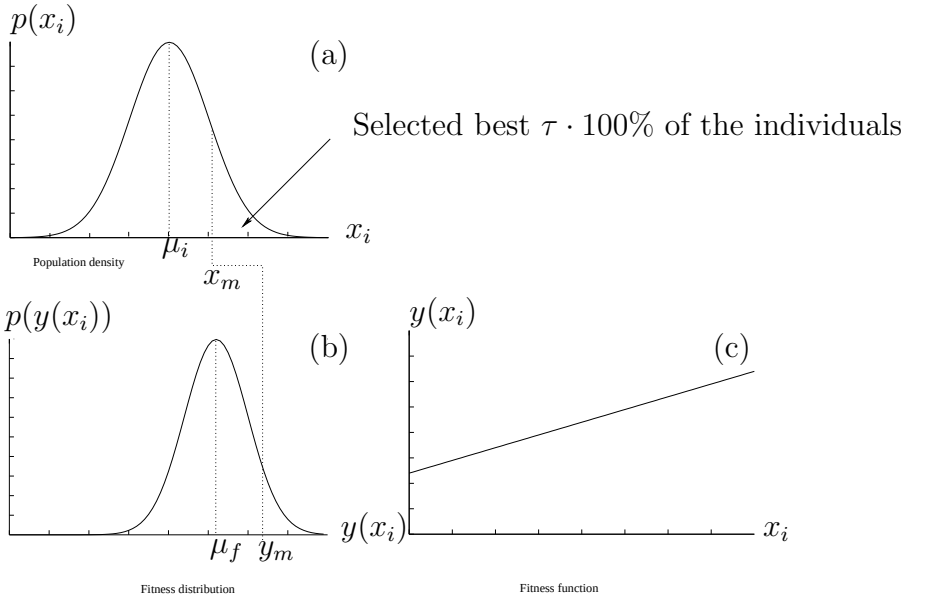


Figure 7.1.: Impact of truncation selection of the best $\tau \cdot 100\%$ individuals on the fitness distribution (b) and the population density (a). We assume an increasing fitness function (c). The fitness distribution is truncated from the left in y_m. The population distribution is truncated from the left in the corresponding point x_m.

The specific structure of the fitness function allows us a decomposition. UMDA$_c$ factorizes over n univariate normals. The fitness function consists of a sum of n univariate monotonous functions.

Thus, we can reduce our analysis to the analysis of one single $g_i(x_i)$. We develop mathematical expressions that model how μ_i^t and σ_i^t change over time. More specifically, we are interested in μ_i^{t+1} and σ_i^{t+1} given the corresponding parameter values at generation t.

Furthermore, we analyze population statistics for a specific generation and the limit behavior of UMDA$_c$. This means we investigate the convergence of μ_i^t and σ_i^t for $t \to \infty$. These calculations are made under the assumption of an infinite population size. For finite samples, they can be seen as an approximation. Here, appropriate order statistics might be used.

We now analyze the truncation selection step in presence of fitness functions of type (7.2). Due to the structure of UMDA$_c$'s probabilistic model and the structure of $f(\boldsymbol{x})$, we can decompose the fitness function and analyze the behavior of each μ_i and σ_i independently.

As we have seen above, we can simplify our approach even further and replace all $g_i(x_i)$ by linear functions of the form $y_i(x_i) = a_i \cdot x_i + b_i$ and study how truncation selection influences the population statistics.

In UMDA$_c$, new candidate solutions are generated by sampling new individuals x_i from a normal distribution with mean μ_i^t and variance $(\sigma_i^t)^2$. The fitness y_i is obtained from a linear function $y_i = a_i \cdot x_i + b_i$. As the $\boldsymbol{x}$ are realizations of a random variable, the fitness can be considered a random variable. The distribution of the fitness Y can be expressed in terms of a normal random variable with mean μ_f and variance σ_f^2:

$$\begin{aligned} \mu_f &= a_i \cdot \mu_i + b_i \\ \sigma_f^2 &= a_i^2 \cdot (\sigma_i^t)^2 \end{aligned} \tag{7.3}$$

By truncation selection, the best $\tau \cdot 100\%$ of the individuals are selected. Thus, all individuals with a fitness larger than a fitness minimum of y_m are selected. Their probabilities sum up to τ. y_m can be obtained from the quantile function of the normal distribution as follows:

$$\begin{aligned} y_m &= \Phi^{-1}_{\mu_f,\sigma_f}(1-\tau) \\ &= \Phi^{-1}(1-\tau) \cdot a_i\sigma_i^t + a_i\mu_i^t + b_i \end{aligned}$$

Statistically, the fitness distribution is truncated from below at y_m. We refer to y_m as the fitness truncation point. Now, we are interested in the individual x_m that corresponds to the fitness value y_m. All individuals $x_i > x_m$ are selected. We call x_m the corresponding population truncation point. This is illustrated graphically in Figure 7.1. The fitness truncation point y_m can be transformed into the population truncation x_m as follows.

$$\begin{aligned} x_m &= y^{-1}(y_m) \\ &= \frac{(y_m - b_i)}{a_i} \\ &= \frac{\Phi^{-1}(1-\tau) \cdot a_i\sigma_i^t + a_i\mu_i^t + b_i - b_i}{a_i} \qquad a_i \neq 0 \\ &= \Phi^{-1}_{\mu_i^t,\sigma_i^t}(1-\tau) \end{aligned} \tag{7.4}$$

It is interesting to see that obviously the truncation point is independent from a_i and $b_i (a_i > 0)$. Put differently, no matter which linear function with $a_i > 0$ we choose, the population truncation point remains the same. Thus, the effect of selection is independent from a_i and b_i, for all $a_i > 0$. No matter how these parameters are chosen, the same individuals are selected.

As we have seen in this section, the selected individuals are identical for all linear functions $y_i(x_i)$, where $a_i > 0$. Furthermore, the population truncation point solely relies on statistical parameters of the population. Thus, the selected individuals can be obtained from the population statistics directly without con-

sidering the fitness landscape.

7.3. Mean dynamics

We have seen that the effect of truncation selection is identical for all linear functions y_i with positive gradient. We have also shown, that for predicting the behavior of UMDA$_c$ under our assumptions we do not need to model the distribution of the fitness. The change in population statistics can be obtained from the population statistics directly. In this section, we derive mathematical expressions for the change of the population mean from generation t to generation $t+1$.

We model selection by truncation of the normally distributed population density. Assume that a population is distributed with mean μ_i^t and standard deviation σ_i^t. The fitness values of the individuals are calculated and the best $\tau \cdot 100\%$ individuals are selected. This equals a left-truncation of the normal distribution in x_m.

To do this, we first summarize results from econometric literature on the truncated normal distribution (see Greene (2003), appendix). We start with presenting the moments of a doubly truncated normal distribution. A doubly truncated normal distribution with mean μ and variance σ^2, where $x_a < x < x_b$ can be modeled as a conditional density where $x \in A = (x_a, x_b)$ and $-\infty < x_a < x_b < +\infty$.

$$f(X|X \in A) = \frac{\frac{1}{\sigma}\phi\left(\frac{x-\mu}{\sigma}\right)}{\Phi\left(\frac{x_b-\mu}{\sigma}\right) - \Phi\left(\frac{x_a-\mu}{\sigma}\right)} \tag{7.5}$$

The moment generating function of this distribution is:

$$\begin{aligned} m(t) &= E(e^{tX}|X \in A) \\ &= e^{\mu t+\sigma^2 t^2/2} \cdot \frac{\Phi\left(\frac{x_b-\mu}{\sigma} - \sigma t\right) - \Phi\left(\frac{x_a-\mu}{\sigma} - \sigma t\right)}{\Phi\left(\frac{x_b-\mu}{\sigma}\right) - \Phi\left(\frac{x_a-\mu}{\sigma}\right)} \end{aligned} \tag{7.6}$$

From the moment generating function, we can derive the statistical moments of the distribution. We are interested in the mean. It is

$$\begin{aligned} E(X|X \in A) &= m'(t)|_{t=0} \\ &= \mu - \sigma \cdot \frac{\phi\left(\frac{x_b-\mu}{\sigma}\right) - \phi\left(\frac{x_a-\mu}{\sigma}\right)}{\Phi\left(\frac{x_b-\mu}{\sigma}\right) - \Phi\left(\frac{x_a-\mu}{\sigma}\right)}. \end{aligned} \tag{7.7}$$

We are not interested in the mean of a doubly truncated normal distribution, but in the mean of a left-truncated normal distribution. Thus, we now let x_b tend to

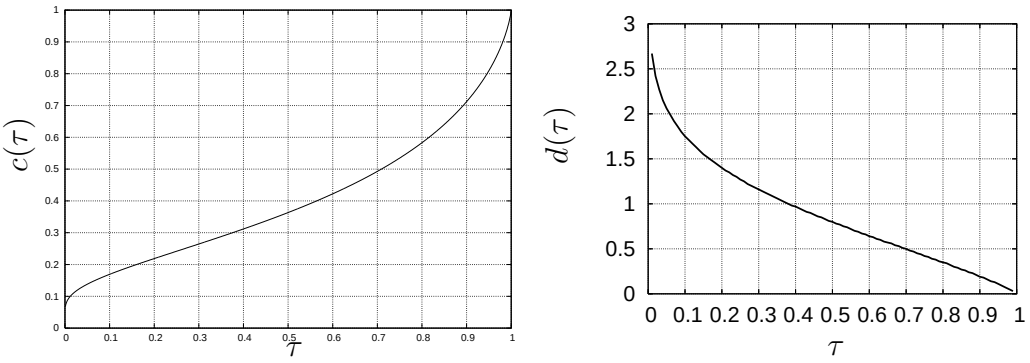


Figure 7.2.: Illustration of $c(\tau)$ and $d(\tau)$

infinity. This results in:

$$E(X|X > x_a) = \mu + \sigma \cdot \frac{\phi(\frac{x_a-\mu}{\sigma})}{\Phi(\frac{x_a-\mu}{\sigma})} \tag{7.8}$$

From Section 7.2 we know that $x_a = x_m = \Phi^{-1}_{\mu_i^t,\sigma_i^t}(1-\tau)$. Inserting and rearranging leads to

$$\begin{aligned} \mu_i^{t+1} &= E(X|X > x_m) \\ &= \mu_i^t + \sigma_i^t \cdot \frac{\phi\left(\Phi^{-1}(\tau)\right)}{1-\tau} \\ &= \mu_i^t + \sigma_i^t \cdot d(\tau), \qquad \text{where } d(\tau) = \frac{\phi\left(\Phi^{-1}(\tau)\right)}{1-\tau} \end{aligned} \tag{7.9}$$

Note that the mean of the population after applying truncation selection can now be easily computed. The factor $d(\tau)$ is illustrated in Figure 7.2. It can be seen that for $\tau \to 1$ the factor $d(\tau)$ converges to 0 leaving the mean of the population unchanged in $t+1$.

7.4. Variance dynamics

Again, we model truncation selection by truncation of the normally distributed population density. Therefore, we again make use of the moment generating

function as in Section 7.3. We let $x_b \to \infty$. Finally, we get:

$$\begin{aligned} Var(X|X > x_m) &= E(X^2|X > x_m) - E(X|X > x_m)^2 \\ &= \sigma^2 \cdot \left\{ 1 + \frac{\frac{x_m-\mu}{\sigma} \cdot \phi\left(\frac{x_m-\mu}{\sigma}\right)}{1 - \Phi\left(\frac{x_m-\mu}{\sigma}\right)} \right. \\ &\qquad \left. - \left[\frac{\phi\left(\frac{x_m-\mu}{\sigma}\right)}{1 - \Phi\left(\frac{x_m-\mu}{\sigma}\right)} \right]^2 \right\} \end{aligned} \tag{7.10}$$

We use this equation in the context of our model by assigning appropriate indices, inserting x_m, simplifying, and rearranging. This leads us to:

$$\begin{aligned} (\sigma_i^{t+1})^2 &= (\sigma_i^t)^2 \cdot \left\{ 1 + \frac{\Phi^{-1}(1-\tau)\phi\left(\Phi^{-1}(\tau)\right)}{\tau} - \right. \\ &\qquad \left. \left[\frac{\phi\left(\Phi^{-1}(\tau)\right)}{\tau} \right]^2 \right\} = (\sigma_i^t)^2 \cdot c(\tau) \\ \text{where } c(\tau) &= \left\{ 1 + \frac{\Phi^{-1}(1-\tau)\phi\left(\Phi^{-1}(\tau)\right)}{\tau} - \right. \\ &\qquad \left. \left[\frac{\phi\left(\Phi^{-1}(\tau)\right)}{\tau} \right]^2 \right\} \end{aligned} \tag{7.11}$$

Now, we can compute the population variance in $t+1$, given the population variance in t. The factor $c(\tau)$ is plotted in Figure 7.2. It can be seen, that if $\tau \to 1$, the factor $c(\tau)$ converges to 1, leaving the variance in generation $t+1$ unchanged.

7.5. Population statistics in generation t

The last two subsections examined the change of the population statistics from one generation to the next generation.

Now, we calculate how the population mean and variance depend on t. To obtain the corresponding population statistics, we sum up the iterative formula that have been developed in 7.3 and 7.4.

Doing this we get the following result for the mean after some calculations. In generation $t > 0$, the mean μ_i^t can be computed as:

$$\mu_i^t = \mu_i^0 + \sigma_i^0 \cdot d(\tau) \cdot \sum_{i=1}^{t} \sqrt{c(\tau)^{i-1}} \tag{7.12}$$

Similarly, in generation t ,the variance σ_t^2 can be computed as:

$$(\sigma_i^t)^2 = (\sigma_i^0)^2 \cdot c(\tau)^t \tag{7.13}$$

7.6. Convergence of population statistics for $t \to \infty$

In this section, we analyze convergence of UMDA$_c$. That means that we analyze how the population statistics develop over time, assuming that $t \to \infty$.

First, we consider the mean. Therefore, we make use of (7.12). Note that the sum is the only part of the expression that depends on t. This leads us to:

$$\lim_{t\to\infty} \mu_i^t = \mu_i^0 + \sigma_i^0 \cdot d(\tau) \cdot \underbrace{\lim_{t\to\infty} \sum_{k=1}^{t} \left[\sqrt{c(\tau)^{(k-1)}}\right]}_{\text{infinite geometric series}} = \tag{7.14}$$

$$\lim_{t\to\infty} \mu_i^t = \mu_i^0 + \sigma_i^0 \cdot d(\tau) \frac{1}{1-\sqrt{c(\tau)}} \tag{7.15}$$

This expression allows us to compute the maximum distance, that UMDA$_c$'s mean will move across the search space for a given selection intensity of $\tau \cdot 100\%$ and monotonous fitness functions.

Now, we consider the variance. We make use of (7.13) and let t tend to infinity. Note that $0 < c(\tau) < 1$. This leads to

$$\begin{aligned} \lim_{t\to\infty} (\sigma_i^t)^2 &= \lim_{t\to\infty} \left[(\sigma_i^0)^2 \cdot c(\tau)^t\right] \\ &= 0 \end{aligned} \tag{7.16}$$

Thus, the variance converges towards 0.

In the previous paragraphs, we derived expressions to describe the behavior of the UMDA$_c$ algorithm with truncation selection on monotonous functions.

We have seen that the algorithm converges since the population variance converges towards 0. The maximal distance that the mean of the population can move across the search space is bounded. This distance solely depends on

- the mean of the first population,
- the variance of the first population,
- and the selection intensity τ.

This has some important effects on the behavior of UMDA$_c$. First, if the optimum lies outside this maximal distance, the algorithm can not find it and one will experience premature convergence. Furthermore, the first population is usually sampled uniformly in the space of feasible solutions. The exploration of the search space relies on density estimation and sampling. However, by choosing the amount of individuals that are selected, one can adjust the maximal distance that the mean of the population will move. One needs to be careful when choosing the selection intensity τ. By wrongly setting τ, the algorithm might not even be able to sample all feasible points. In essence, these results agree with the results from González et al. (2002). They show that for simple linear functions the same problems exist when the tournament selection scheme is used.

A simple solution to combat premature convergence would be to increase the sampling variance beyond its maximum likelihood estimate. This could combat the variance decrease and allow the algorithm to traverse slope-like regions of the search space.

8. Optimal sampling variances and runtime

According to the major lesson learned from the last chapter, using the maximum likelihood variances to sample offspring from can easily lead to premature convergence. This is due to the fact that the variance decreases towards zero. In practical optimization, this happens too fast. Variance enlargement appears to be crucial if a continuous EDA traverses a slope-like region of the search space. Section 8.1 aims at answering the fundamental question, whether the sampling variance can be set arbitrarily high, or whether values exist that should preferably be chosen when an EDA is in phase (2) of the search. Furthermore, it is still an open question how fast an EDA can solve the sphere function to a given precision. This is relevant for EDA in phase (3) of the search and will interest us in Section 8.2.

8.1. Optimal sampling variances

In this section, we seek an answer on how to set the sampling variance. To this end, assume the simplified case that the one-dimensional sphere function $f(x) = x^2$ should be minimized to a value to reach v. All solutions x that lie inside an optimal region R have a fitness smaller than v. For the one-dimensional case, $R = [-\sqrt{v}; +\sqrt{v}]$. Consequently, we seek to find a variance that maximizes the chance to sample candidate solutions inside R. The success ratio s measures the overall probability that a solution sampled from a one-dimensional normal distribution with mean μ and variance σ^2 lies inside an optimal region $R = [a, b]$, with $\mu < a < b$, and is defined as

$$s(\mu, \sigma^2, a, b) = \Phi_{\mu,\sigma^2}(b) - \Phi_{\mu,\sigma^2}(a).$$

Without loss of generality we set $\mu = 0$, a and b are known parameters. The aim is to find a sampling variance $(\sigma^2)^\star$ that maximizes s:

$$(\sigma^2)^\star = \arg\max_{\sigma^2 \in R^+} s(0, \sigma^2, a, b)$$

The first order derivative of $s(\mu, \sigma^2, a, b)$ with respect to σ is given as

$$\frac{\mathrm{d}s}{\mathrm{d}\sigma} = -\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{b^2}{2\sigma^2}} \cdot b + \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{a^2}{2\sigma^2}} \cdot a. \tag{8.1}$$

(8.1) has two roots, one of which is infeasible due to negativity. The positive root

$$(\sigma^2)^\star = \left(\frac{b^2 - a^2}{2 \ln \frac{b}{a}} \right)$$

is a possible feasible maximizer for s. The second order derivative of $s(\mu, \sigma^2, a, b)$ with respect to σ is

$$\frac{\mathrm{d}^2 s(\mu, \sigma, a, b)}{\mathrm{d}\sigma^2} = \frac{e^{-\frac{b^2}{2\sigma^2}}}{\sqrt{\pi}} \cdot \left(\frac{\sqrt{2}}{\sigma^3} - \frac{b^3}{\sqrt{2}} \right) - \frac{e^{-\frac{a^2}{2\sigma^2}}}{\sqrt{\pi}} \cdot \left(\frac{\sqrt{2}}{\sigma^3} - \frac{a^3}{\sqrt{2}} \right). \tag{8.2}$$

It can easily be shown, that (8.2) is $< 0 \ \forall \ a < b$. Thus, $(\sigma^2)^\star$ is the unique maximizer for the success probability. If a converges towards the mean $\mu = 0$, then $(\sigma^2)^\star \to 0$.

The existence of a unique maximizer for the success probability is an interesting and novel result with some important consequences for EDA design. If R is known, a sampling variance of $(\sigma^\star)^2$ maximizes the number of solutions that are sampled in R and hence maximizes convergence speed. If the used sampling variance deviates from $(\sigma^\star)^2$, less individuals will be sampled in R. The result applies only for one-dimensional search spaces, and we do not provide an extension to multi-dimensional spaces. A starting point for this analysis can be elliptically truncated multivariate normal distributions and their moments as presented in Kotz et al. (2000). We conjecture, that a similar maximizer exists for the multi-dimensional case as well, although it is more difficult to obtain.

8.2. Runtime bound

In this section, we analyze phase (3) of the search. The mean is positioned inside the optimal region. Not all solutions are optimal, as the success ratio depends also on the sampling variance. We derive a lower bound on the number of generations that a continuous EDA utilizing truncation selection needs in order to solve the sphere function to a given precision. Provided parameters are an initial variance σ_0^2 for each dimension of the normal distribution and a value to reach v. We consider the simpler one-dimensional case first and extend the results to n dimensions.

8.2.1. Runtime on x^2

The sphere function in a single dimension is $f(x) = x^2$. We consider the case that the EDA has already located the optimal solution $x^\star = 0$ and that the mean

μ is $\mu = x^\star$. Under an infinite population size, μ will not move away from $x^\star$ in an EDA run. Truncation selection selects solutions point-symmetrically to $x^\star$. The consequence is that $\mu_t = 0 \; \forall \; t$.

Change from σ_t^2 to σ_{t+1}^2

Given a variance in period t denoted by σ_t^2 and the fraction of selected individuals τ, we seek to derive σ_{t+1}^2 – the variance after truncation selection. Since $\mu = x^\star = 0$, all individuals that lie inside the unique interval $[-w, w]$ satisfying

$$\int_{-w}^{w} \phi_{0,\sigma_t^2}(x)\mathrm{d}x = \tau$$

are selected. Selection equals a double truncation of the normal distribution. The variance of a doubly truncated normal distribution can be expressed in simple terms for the special case of truncation limits $A, B, A < B$ that are symmetric around the mean, cf. Johnson et al. (1994) p. 158. If $A - \mu = -(B - \mu) = -\kappa\sigma$, then the mean of the truncated distribution is, again, μ. The variance $\sigma^{2'}$ of the truncated normal is

$$\sigma^{2'} = \sigma^2 \cdot \left(1 - \frac{2\kappa\phi(\kappa)}{2\Phi(\kappa) - 1}\right). \tag{8.3}$$

This special case applies here. The upper bound B can be determined by

$$B = \Phi^{-1}(0.5 + 0.5\tau) = \kappa.$$

The variance after selection can thus be written as

$$\begin{aligned}
\sigma_{t+1}^2 &= \sigma_t^2 \left(1 - \frac{2\Phi^{-1}(0.5 + 0.5\tau)\phi\left(\Phi^{-1}(0.5 + 0.5\tau)\right)}{\tau}\right) \\
&= \sigma_t^2 \cdot b(\tau), \text{with} \\
b(\tau) &= \left(1 - \frac{2\Phi^{-1}(0.5 + 0.5\tau)\phi\left(\Phi^{-1}(0.5 + 0.5\tau)\right)}{\tau}\right).
\end{aligned}$$

Thus, the variance is decreased by a constant factor that solely depends on the selection intensity. The term $b(\tau)$ can easily be computed numerically.

Variance in generation t

It is straightforward to derive the variance in a generation t if we know an initial variance σ_0^2. As the variance is decreased by $b(\tau)$ in each generation, the variance in generation t is

$$\sigma_t^2 = \sigma_0^2 \cdot b(\tau)^t. \tag{8.4}$$

Runtime

We define the runtime as the number of generations required until the variance has decreased so far, that solutions with a fitness that is smaller than a value to reach v are sampled with a probability of at least 99.5%. Note that other probabilisties can equally be chosen.

It is required that $|x| < \sqrt{v}$, for x to be optimal. The above chance constraint can be expressed as

$$P(x \in [-\sqrt{v}; \sqrt{v}]) \geq 0.995. \tag{8.5}$$

It is known that 99.5% of the mass of the normal distribution lies within its 3σ-quantile. Thus, we can rewrite (8.5) as

$$\begin{aligned} 3\sigma &< \sqrt{v} \\ \Leftrightarrow \sigma^2 &< \frac{v}{9}. \end{aligned}$$

As soon as the variance has decreased to a value smaller than $\frac{v}{9}$, optimal solutions are sampled with a probability of at least 99.5%. Using (8.4) , this will be the case, if

$$\sigma_0^2 b(\tau)^t \leq \frac{v}{9}.$$

Solving for t yields a runtime of

$$t \geq \frac{\log \frac{v}{9\sigma_0^2}}{\log b(\tau)}. \tag{8.6}$$

Equation (8.6) gives a lower bound on the runtime on the one-dimensional sphere function that depends on a value to reach, an initial variance, and the selection intensity.

8.2.2. Runtime on the l-dimensional sphere function

The l-dimensional sphere function is point-symmetric to 0 and has a unique global optimum at $\boldsymbol{x}^\star = 0$ with $f(\boldsymbol{x}^\star) = 0$. The result from Section 8.2.1 is generalized to l-dimensional spaces in the following. Therefore, $\boldsymbol{\mu} = \mathbf{0}$, the reasons being equal to the one-dimensional case.

The multivariate normal distribution

The l-dimensional normal distribution is given by a mean vector $\boldsymbol{\mu}$ and a positive semi-definite covariance matrix Σ of order $(l \times l)$. The eigenvectors of Σ are denoted by $\boldsymbol{e}_i$, $i = 1, 2, \ldots, l$. The eigenvalues of Σ are denoted by $\boldsymbol{\lambda}_i$, $i = 1, 2, \ldots, l$.

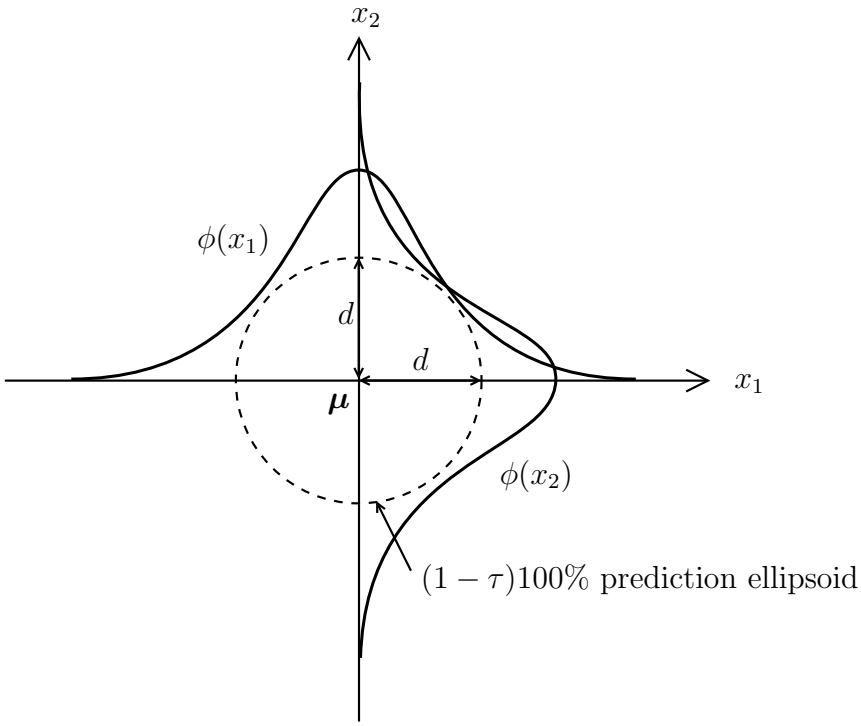


Figure 8.1.: Univariate factorization of a two-dimensional normal pdf, prediction ellipsoid and half axes with lengths d. $\phi(x_1)$ and $\phi(x_2)$ denote the normal pdfs associated with x_1 and x_2.

We are especially interested in some geometric properties of the multivariate normal. Points of equal density lie on hyper-ellipsoids. The half-axes of the ellipsoids point in direction of the eigenvectors of the normal distribution. The eigenvalues relate directly to the length of the associated half-axes. The covariances induce rotation of the ellipsoids around the mean. If all covariances are 0, the axis of the ellipsoids point exactly into the directions of the main axes of the reference (coordinate) system. Prediction ellipsoids are the smallest ellipsoidal regions in l-dimensional space that are centered around the mean and contain a certain percentage of the mass of the multivariate normal. Further details can be found in Kotz et al. (2000).

Change from Σ_t to Σ_{t+1}

In order to solve the l-dimensional sphere function, the utilized EDA estimates and samples an l-dimensional normal distribution. It is assumed that in the estimation process no superfluous dependencies between the l random variables are introduced. This means, that only dependencies that result from the function that is optimized are introduced and sampling as well as estimation errors are neglected. Hence, the estimated model is a univariate factorization. This product of l univariate normals matches the separable structure of the sphere function perfectly. The general idea of our approach is that the modifications of the covariance matrix due to selection can be expressed fully in terms of modifications in the univariate normals. First it is analyzed which solutions in l-dimensional space are selected. Then, the impact of selection is modeled in one-dimensional space. This allows to reuse the basic idea from Section 8.2.1.

It is known that $(1-\alpha)\cdot 100\%$ of the mass of the multivariate normal lies within the so-called $(1-\alpha)$ prediction ellipsoid. This ellipsoid has half-axes with lengths of $d_i = \sqrt{\lambda_i \chi^2_{l,\alpha}}, i = 1, 2, \ldots, l$. Recall, that $\chi^2_{l,\alpha}$ denotes the $(1-\alpha)$ quantile of the chi-square distribution with l degrees of freedom. Truncation selection selects all individuals that lie in a prediction ellipsoid that covers exactly $\tau \cdot 100\%$ of the density. We are interested in the inner region of the ellipsoid, so $\alpha = 1-\tau$. Since selection is point-symmetric to the optimal solution all eigenvalues have equal values $\lambda_i = \lambda \; \forall \; i = 1, 2, \ldots, l$. The prediction ellipsoid that contains all selected individuals thus has axes with half lengths $d = \sqrt{\lambda \chi^2_{l,1-\tau}}$. If the probabilistic model consists of a factorization of univariate normals, the eigenvalues λ denote the variances into this direction. Hence the half-length d of the axes of the prediction ellipsoids covering the selected solutions is

$$d = \sigma \cdot \sqrt{\chi^2_{l,1-\tau}}. \tag{8.7}$$

A graphical illustration of this result is depicted in Figure 8.1.

Knowing d we can reuse the approach from Section 8.2.1 to derive the variance in the next generation. Utilizing (8.3) leads to a variance after selection

$$\sigma_{t+1}^2 = \sigma_t^2 \cdot \left(1 - \frac{2\chi_{l,1-\tau}^2 \phi\left(\sqrt{\chi_{l,1-\tau}^2}\right)}{2\Phi\left(\sqrt{\chi_{l,1-\tau}^2}\right) - 1}\right)$$

$$= \sigma_t^2 \cdot \chi(l,\tau), \text{with} \chi(l,\tau) \qquad = \left(1 - \frac{2\chi_{l,1-\tau}^2 \phi\left(\sqrt{\chi_{l,1-\tau}^2}\right)}{2\Phi\left(\sqrt{\chi_{l,1-\tau}^2}\right) - 1}\right).$$

Like in previous calculations of the variance after selection, the variance is reduced by a constant factor. The factor $\chi(l,\tau)$ solely depends on τ and l and can be computed numerically.

Variance in generation t

Given an initial variance σ_0^2 in each dimension and the number of dimensions n, the variance in generation t can be computed as

$$\sigma_t^2 = \sigma_0^2 \cdot \chi(l,\tau)^t.$$

Runtime

Section 8.2.2 showed that the overall modification of the covariance matrix can be expressed also through the modifications of the l variances assuming a univariate factorization and an initial isotropic normal distribution. In order to derive the runtime of a continuous EDA on the l-dimensional sphere function, we define a target standard deviation that, if used to sample from, will cause 99.5% of the solutions to have a fitness value smaller than a given value to reach v. Then we analyze how many generations selection must reduce the variance in order to reach this target value.

Optimal solutions are solutions whose fitness is smaller than v. For solutions to be optimal it is required that

$$\sum_{i=1}^{l} x_i^2 \leq v. \tag{8.8}$$

Under infinite population sizes the EDA behaves identically for every dimension. This has allowed the dimensionality reduction in the previous section and allows

to rewrite (8.8) as

$$lx^2 \leq v$$
$$\Leftrightarrow x \leq \sqrt{\frac{v}{l}}.$$

In order to sample 99.5% optimal solutions, it is required that

$$3\sigma \leq \sqrt{\frac{v}{l}}.$$

Inserting the general variance in generation t leads to

$$3\sigma_0 \cdot \chi(l,\tau)^{\frac{t}{2}} \leq \sqrt{\frac{v}{l}},$$

which can be solved for t. The necessary number of generations t is at least

$$t > 2\frac{\log \frac{\sqrt{\frac{v}{l}}}{3\sigma_0}}{\log \chi(l,\tau)}. \tag{8.9}$$

The runtime depends on an initial variance, the selection intensity, the value to reach, and the number of dimensions.

It can easily be seen that - once search is in phase (3)- reducing the variance is preferable for reducing the runtime.

8.3. Summary and conclusion

How do continuous EDA really work? Much of the currently available results are experimental. A thorough understanding of continuous EDA can only be achieved on the basis of formal models. Taking together the current literature, many important questions are still unanswered. One of the most important questions is how the parameters of the probabilistic models (e.g., the covariance matrix and the mean vector) change over time due to selection. Such results are difficult to obtain if the underlying fitness landscape is complex. We have concentrated on the sphere function and have artificially decomposed the convergence process into the following three phases.

1. The mean of the normal distribution is far from the optimum. The EDA traverses a region that has a slope-like function. The optimum is not sampled with significant probability.
2. Selection has shifted the mean towards the optimum. A significant portion

of the sampled solutions lie in the optimal region, but the mean is still outside the optimal region.

3. The mean has moved onto the optimum and is relatively stable.

All three phases are characterized through ML-variance trajectories, that is a series of subsequent variances modified over generations through selection. In the first phase, variances estimated by maximum-likelihood estimators have been proven to lead to premature convergence. As a consequence, variance enlargement was introduced in Ocenasek et al. (2004), Yuan and Gallagher (2005) and Bosman and Grahl (2008). This has lead to trajectories of sampling variances that are not equal to the estimated ones. In order to traverse slopes, a maximal increase of the ML-variance is beneficial as it increases progress.

It is important to know, whether this increase can come at a price in later phases of optimization. We have shown in Section 8 that too high a variance can indeed slow down progress if the optimal solution is coming "into sight" and is sampled with significant chance. For the one-dimensional case we have proved the existence of a sampling variance that is the unique maximizer of the success ratio (recall, that the success ratio was defined as the proportion of solutions sampled in the optimal region). This optimal sampling variance decreases with the distance between the mean and the region, and converges towards zero for the extreme case that the mean has reached the border of the region. We have not provided an extension of this result to the general multi-dimensional case, but it appears highly likely that a similar result applies, although it might be more difficult to obtain. Obviously, if the sampling variance that an EDA uses is very close or equal to the optimal sampling variance, progress of the search is maximized. If the sampling variance is too high, or too low, fewer sampled individuals are optimal.

In the third phase, selection has moved the mean onto the optimum and it is relatively stable. Until now, it was unknown how fast a continuous EDA can contract the distribution around a point of interest. We provided in Section 8.2.2 such a runtime result. The number of generations required until a value to reach is reliably sampled on the sphere function can be computed. By multiplying with a sensible estimate for the population size, the number of fitness evaluations can be approximated easily. A deeper analysis of these results will not only be interesting on its own, but also open the door for a principled comparison of continuous EDA with, e.g., evolution strategies.

From the results on phase (1) dynamics, and the results on phase (3) dynamics, we conjecture that the variance will decrease exponentially fast also if an EDA is in phase (2). This would mean that the variance goes down steadily over time throughout an EDA run. This leads to premature convergence on slope like regions of the search space. Furthermore, it is highly unlikely that such a

decrease matches the optimal variances required for maximal progress in phase (2). However, it is a sensible approach in phase (3).

The results are interesting, as they are substantially different to the ones obtained for discrete domain. An in-depth discussion of the differences as well a novel variance scaling policy is presented in the next chapter. The three phases are assumed as independent. In an EDA run, the EDA will move from one phase to another and it is interesting to identify the "current" phase on the fly. A simple method that strives answer this question will be introduced in the next chapter as well.

9. Matching search bias and problem structure: CT-AVS-IDEA

9.1. Introduction

This chapter proposes a principled contribution to explaining what goes wrong in the adaptation of EDA from the discrete domain to the continuous domain and to solving the problem of premature convergence. Therefore, it is assessed which requirements a probability distribution has to meet in order to function properly as a search distribution in EDA. We argue that there is a fundamental and systematic difference between the discrete and the continuous domain. Generally speaking, in order to build efficient optimizers using the EDA principle, the induced bias in the form of the estimated probability distribution has to fit to the structure of the problem at hand.

Thereby, a central topic is to assess the discrepancies between the concept of problem structure in the discrete and continuous domain and to assess to which extent the probabilistic search bias can fit the problem structure in the continuous domain. We indicate that indeed compared to the discrete case there are additional issues that need to be addressed in the design of the continuous EDA to decrease the probability of failure. We also present a simple remedy to meet with the additional issues we identify and show on the basis of experimental results that consequently the optimization performance of the continuous EDA indeed improves substantially.

The remainder of this chapter is organized as follows. In Section 9.2, we transfer lessons learned from the discrete domain to the continuous domain. The notion of inductive search bias, problem structure and how it relates to the use of distributions in EDA is discussed in Section 9.2.1. Henceforth, these concepts are exemplified for discrete EDA in Section 9.2.2 and for continuous EDA in Section 9.2.3. We also point out how the main lessons learned from the discrete domain can be transferred to the continuous domain. We show that a more careful interpretation is required of these lessons for the proper design of continuous EDA. We illustrate by experiments how and why continuous EDA can indeed fail. Afterwards, in Section 9.3, we propose a straightforward remedy with virtually no additional computational overhead. Integrating it into a continuous EDA yields the correlation-triggered adaptive variance scaling IDEA (CT-AVS-IDEA). It is compared to a continuous EDA and the state of the art in continuous evolutionary optimization in Section 9.4. We conclude this chapter in Section 9.5.

9.2. Adapting discrete EDA to continuous EDA

9.2.1. Matching inductive search bias and problem structure

In general, for an optimization algorithm to be competent in solving a certain optimization problem, the search bias of the optimization algorithm has to fit the structure of the problem. The search bias of EDA, exemplified by the probability distribution used, is inductive as it is learned during optimization. Now, if it is possible to approximate the probability distribution over the solution space that assigns a uniform probability distribution over all solutions with a quality at least as good as that of the worst selected individual, a highly efficient EDA can be constructed, see Rastegar and Meybodi (2005). This EDA fine-tunes the probability distribution each generation to represent ever more precisely and selectively the best solutions in the search space. For EDA, therefore, the following two prerequisites are of specific importance:

1. Adequacy of the class of probability distribution
   The probability distribution must be able to assign solutions that have a certain minimal quality, i.e. solutions that have specific properties, a high probability density. In other words, the *capacity* of the probability distribution must be adequate.
2. Competence of the estimation procedure
   Even if the capacity of the class of probability distribution used is adequate, proper exploitation of the structure of the optimization problem is only guaranteed if the estimation procedure is actually capable of configuring the parameters of the probability distribution in such a way that the high probability densities are actually assigned to solutions of a certain minimal quality. In other words, the estimation procedure must be *competent*.

Finally, it should be noted that for efficiency, an additional prerequisite is that the estimation procedure is efficient (i.e., of low–order asymptotic algorithmic complexity) in addition to being competent.

9.2.2. Discrete EDA

Inductive search bias

Probability distributions for discrete spaces assign probabilities to specific settings of variables. Hence, any probability distribution can be expressed, ensuring an adequate capacity. By factorizing the probability distribution (see Lauritzen (1996); Friedman and Goldszmidt (1996)), not all combinations of settings for all variables need to explicitly be enumerated, but probabilities can be assigned to specific combinations for subsets of variables. Since factorizations are only a

more efficient way of representing the probability distribution, the adequacy of the capacity is not affected. Using frequency counts to estimate the probabilities from data results in maximum–likelihood estimations that reveal statistical dependencies.

Problem structure

In the discrete domain, problem structure refers to a decomposition of the optimization problem into sub-problems of smaller sizes Goldberg (2002). In other words, there are configurations of bits at specific locations, so–called Building Blocks (BB), that contribute significantly to the solution quality when present in a solution. These building blocks are commonly said to form partial solutions to the problem. Moreover, the knowledge of which bit–configurations at what locations cause a significant contribution to the solution quality is commonly referred to as *linkage information*, see Harik and Goldberg (1997).

Matching

The necessity of the joint appearance of configurations of bits causes statistical dependence of random variables when estimating the probability distribution of the configurations of the bits from a set of solutions that were selected on the basis of their quality. Using factorized probability distributions, these statistical dependencies can be modeled. In other words, a discrete EDA can store which configurations of bits should have a large probability of appearing jointly in a good solution because the capacity prerequisite from Section 9.2.1 is met.

It has to be noted however, that in accordance with the degree of the interactions between the bits, simple or more involved factorizations need to be used. If there are no interactions between the bits, meaning that the building block size is one, univariately factorized probability distributions in which each variable is modeled to be statistically independent of each other variable, have proven to be efficient when used in an EDA, see Pelikan and Mühlenbein (1998). In general however the size of the building blocks is larger. As the interactions between the bits get more complex, the possibilities for expressing statistical dependency relations in the probability distributions should increase accordingly, see Thierens (1999) and Bosman and Thierens (1999). To this end, Bayesian factorizations have been found to be a suitable choice, see Pelikan et al. (1999), Mühlenbein and Mahnig (1999) and Pelikan and Goldberg (2003) as they meet capacity prerequisite and in addition, a greedy estimation procedure is often found to meet the competence prerequisite from Section 9.2.1.

Summarizing, in the discrete space the inductive bias of factorized probability distributions based on frequency counts can match the decomposability of the

problem structure. In addition, assuming that the decomposability is of bounded complexity, the greedy estimation procedure that is commonly employed in discrete EDA is both competent and efficient. Consequently, discrete EDA allow for efficient optimization.

9.2.3. Continuous EDA

Inductive search bias

In the continuous domain, it is the contour–lines of the probability distributions that indicate which parts of the search space have a higher probability of being sampled.

Problem structure

Analogous to the inductive search bias, the problem structure in the continuous domain is exemplified by the contour–lines of the function to be optimized.

Matching

To match inductive search bias and problem structure, we thus need to match the contour lines. However, because the contour–lines of the optimization problem can be of virtually any shape, we require the property of universal approximation. However, such universal approximation is computationally intractable. In practice, a continuous EDA will therefore have to rely on tractable probability distributions, such as ones that are based on the normal pdf.

Because in general we cannot assume that the contours of the fitness function can be modeled properly, a problem arises. The concept of statistical dependence no longer corresponds with dependence as imposed by the fitness landscape. For instance when using the normal pdf, after estimating the parameters it might be found that there is no statistical dependence between two variables. However, when observing the actual source for the data, which follows the density contours of the fitness function, the variables may be strongly dependent through non–linear interactions that simply cannot be modeled by the normal pdf. Hence, we can now conclude the following implications for the design of continuous EDA regarding adequacy:

1. Adequacy of the class of probability distribution (continuous domain)
   a) *Adequate class of probability distribution*
      Linkage information in the continuous case only maps perfectly onto statistical dependency information as observed in the factorization of

the probability distribution if the class of probability distribution that is used to perform density estimation with has the capacity to allow for a close modeling of the contours of the fitness landscape.

b) *Inadequate class of probability distribution*
Assume that there is a mismatch between the capacity of the class of probability distribution used for estimation and the contours of the fitness landscape. Then, the modeling of statistical dependencies through factorizing the probability distribution in estimating the distribution from data is a less important and less reliable source of information for inducing the search bias.

From these revised prerequisites it follows that in an EDA based on the normal pdf it appears not to be a promising way to approach probabilistic modeling in the continuous domain with the same goal as in the discrete domain: to focus solely on getting the statistical dependencies right in the estimated model and thereby assume a proper problem decomposition. Indeed, in initial EDA that estimate a Bayesian factorization of the normal distribution using maximum–likelihood estimations, problems of inefficiency were already revealed after performing experiments on a variety of problems with differing problem structures, see Bosman (2003).

The inductive search bias on a slope

It was observed that initial EDA are not capable of exploiting gradient information since density estimation makes no assumption on the source of the data from which to make the estimations. As a result, these EDA were found to be extremely inefficient on problems with strong non–linear interactions between the variables, even in the presence of smooth gradients and unimodality. As a consequence of the approach taken, premature convergence can occur even on very simple functions. This was formally illustrated in Chapter 7 and will be generalized to slope-like regions in this section. Therefore, the function to be optimized is a sphere function, with an EDA initiated on one side of the optimum, representing a slope-like region that must be traversed in order to advance to the optimum.

An experimental illustration of optimization failure is presented in Figure 9.1. The Figure shows the result of using a one–dimensional maximum–likelihood normal EDA to minimize the sphere function. The progression of density estimations is shown in subsequent generations. The solutions are initially in the range $[-10; -5]$. Indeed, even though the function to optimize has a smooth gradient and is unimodal, the EDA is not able to find the optimum because the variance goes to zero too rapidly. The problem caused by lack of generalization is immediately apparent from this Figure.

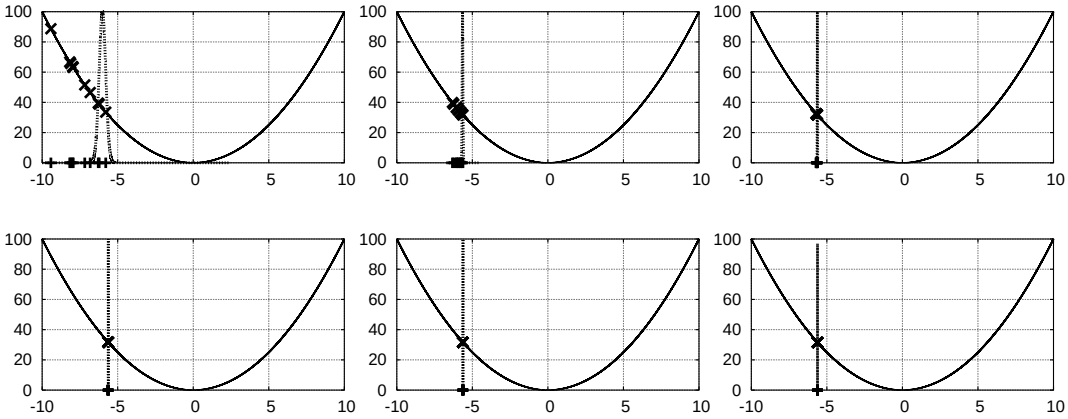


Figure 9.1.: Population and estimated probability distribution (rescaled to fitness range for visualization) in the maximum–likelihood normal EDA in generations 0, 1, 2, 3, 4 and 5 (top–left to bottom–right).

The resulting situation is comparable to the loss of building blocks during a GA run in the discrete domain. Discrete GA theory tells us that in that case the population size should be increased. However, because of the limiting shape of the density function to be estimated we know that increasing the population size will not help to obtain a better approximation of the true density. Hence, the population size will have to increase dramatically to improve the initial quality of solutions. Under the use of elitism, these solutions will then be maintained throughout the run, increasing the eventual possibility of ending up with a distribution of solutions surrounding a peak. However, such increase will be exponential in the number of variables due to the curse of dimensionality. Moreover, if the optimum is simply not contained in the initial range in which samples are available, increasing the population size will not improve the probability of finding the optimum at all.

Concluding, using maximum–likelihood estimations of normal pdfs, the search bias cannot be fit to match the structural properties of slope–like regions in the fitness landscape due to lack of generalization.

The inductive search bias on a peak

Assuming that solutions are distributed nicely surrounding a peak in the landscape, it is evident that the search bias induced by an EDA based on the normal pdf will fit the problem structure well. The reason is that the unimodality of the normal pdf will place the center of mass of the estimated distribution near the

true center of the landscape, increasing the probability of generating solutions near the optimum.

9.3. Adaptive variance scaling and correlation triggering

In the previous chapter we have analyzed how the induced bias of the normal pdf fits to two elementary structures of a continuous search space: slopes and peaks. We found, that the induced search bias cannot be made to fit the structure of a slope well enough to guarantee successful search, whereas it imposes no problem on peaks. Both structures will, however, in general appear during an EDA run. Since we are not interested in making the class of probability distribution more involved in this chapter, the most important question that now arises is how the estimation procedure in the normal EDA should be changed to prevent the identified problems as best possible.

In this section we first introduce a simple technique that modifies the estimation procedure of the normal pdf in a way that makes it more effective when traversing a slope. Subsequently we propose a triggering method that allows to decide during optimization whether the use of this efficiency enhancement is currently appropriate or not.

9.3.1. Adaptive variance scaling

We propose a technique that modifies the estimation procedure of the normal pdf in continuous EDA to make it more reliable when traversing a slope. The smaller the variance is in the estimated probability distribution, the smaller the area of exploration for the EDA. The variance in the normal pdf is explicitly stored in the covariance matrix $\mathbf{\Sigma}$. Hence, a straightforward manner to allow the EDA to increase the area of exploration and thereby increasing the probability of traversing a slope is to enlarge the variance beyond its maximum-likelihood estimate.

Therefore, an adaptive-variance-scaling coefficient c^{AVS} is maintained. Upon drawing new solutions from the probability distribution, the distribution is scaled by c^{AVS}. This means, that the covariance matrix used for sampling the normal pdf is $c^{\text{AVS}}\mathbf{\Sigma}$ instead of just $\mathbf{\Sigma}$. If the best fitness value improves in one generation, then the current size of the variance allows for progress. Hence, a further enlargement of the variance may allow further improvement in the next generation. To fight the variance diminishing effect of selection, the size of c^{AVS} is scaled by $\eta^{\text{INC}} > 1$. If on the other hand the best fitness does not improve, the range of exploration may be too large to be effective and the adaptive variance scaling coefficient should be decreased by a factor $\eta^{\text{DEC}} \in [0, 1]$. For symmetry, we set $\eta^{\text{INC}} = 1/\eta^{\text{DEC}}$.

We bound the magnitude of c^{AVS} from above by a predefined value $c^{\text{AVS-MAX}}$ and from below by a predefined value $c^{\text{AVS-MIN}}$. For symmetry, we set $c^{\text{AVS-MIN}} = 1/c^{\text{AVS-MAX}}$. Moreover, if $c^{\text{AVS}} < c^{\text{AVS-MIN}}$, we set c^{AVS} to $c^{\text{AVS-MAX}}$ in order to stimulate exploration.

An experimental illustration of the normal EDA extended with the adaptive–variance–scaling technique is presented in Figure 9.2. Indeed the EDA is now capable of finding the optimum even though it is outside of the initial sampling range.

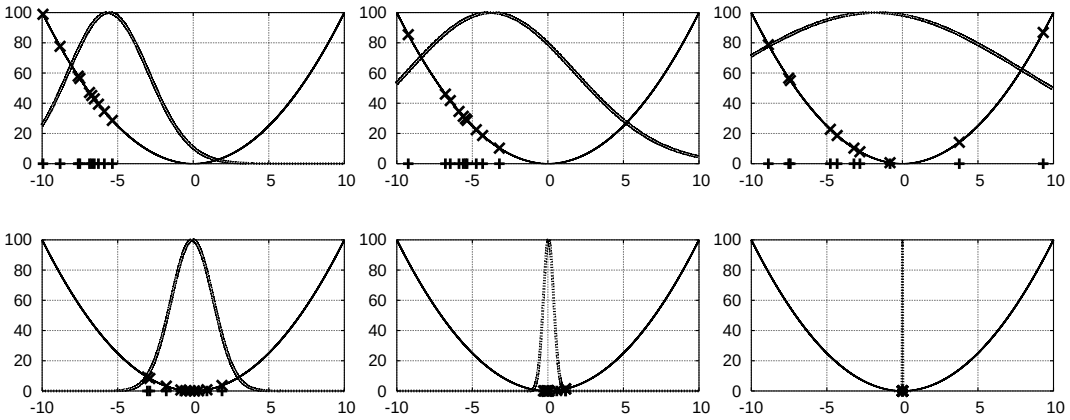


Figure 9.2.: Population and estimated probability distribution (rescaled to fitness range for visualization) in the adaptive–variance–scaling maximum–likelihood normal EDA in generations 0, 1, 2, 4, 8 and 16 (top–left to bottom–right).

9.3.2. Correlation–triggered adaptive–variance-scaling

In the scheme defined in Section 9.3.1 the adaptive–variance–scaling coefficient c^{AVS} increases if a better fitness value is found, i.e. if the EDA is successful in a certain generation. A success does however not always mean that the variance needs to be enlarged. This is especially the case when the center of the normal pdf is close to the optimum. Once this is the case, the induced bias of the normal pdf suffices to guide the search to the optimum. Making the variance larger in such a case will only slow the EDA down as it leads the bias of the algorithm to also explore a larger area around the optimum. Because this essentially makes the EDA less efficient, adaptive variance scaling is to be prevented in such a case. Note that this approach to distinguish between the two situations during the EDA run is actually a test that indicates whether the currently induced search bias

suits the structure of the current search area. If it does, the maximum–likelihood probabilistic modeling of the normal pdf can be used (following the combination of prerequisites 1a and 2). Otherwise, additional means of inducing the search bias may be extremely helpful (following the combination of prerequisites 1b and 2).

To obtain a test of the reliability of using structure identification in continuous EDA by means of maximum–likelihood estimations, the relationship between normal density and fitness of the selected solutions can be exploited. If the selected solutions are centered around a (local) optimum, the density will be strongly correlated with fitness (positively in case of maximization and negatively in case of minimization). The reason is that for the normal distribution the density of a point decreases when it is moved away from the mean. Intuitively, such correlation is desirable since better fitness values get a higher probability of being (re)produced by the EDA. If on the other hand the selected solutions are found to be on a slope, on the one side of the mean the fitness values will be better whereas on the other side of the mean the fitness values will be worse. Hence, a decrease in density is associated with both an increase and a decrease in fitness, effectively decorelating density and fitness.

We propose to base the test for triggering the use of adaptive variance scaling on the ranked correlation coefficient between density and fitness. We use ranked correlation because the most important aspect is that a larger density should be associated with a better fitness value whereas the exact form of the fitness landscape is less important. The results of using this correlation trigger for a slope and for a peak are illustrated in Figure 9.3.

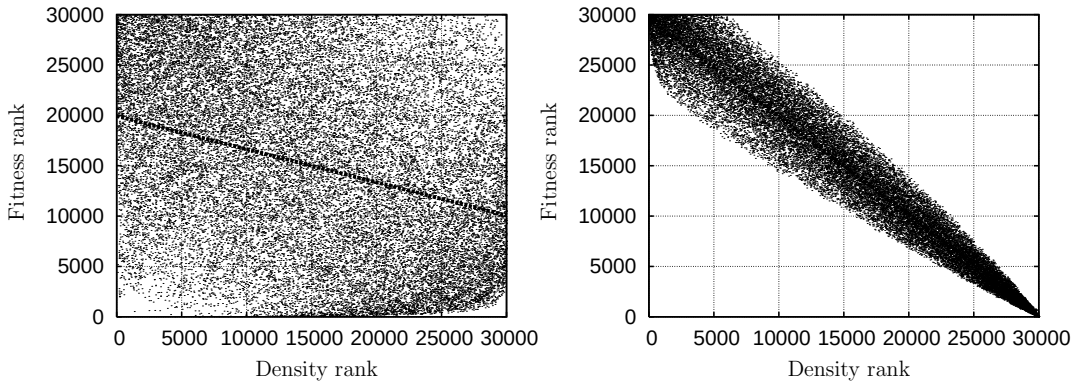


Figure 9.3.: Scatterplots and corresponding regression lines for fitness of the selected solutions versus their density under the estimated normal distribution in the first generation when minimizing the sphere function for $l = 5$. *(Left)* initial range $= [-10, -5]$ ($r = -0.3289859$), *(Right)* initial range $= [-3, 2]$ ($r = -0.9725636$).

We propose to have a threshold value θ^{corr} such that if the value of the correlation coefficient r between the density and the fitness of the selected solutions is at most the value of the threshold, i.e. $\theta^{\text{corr}} \leq r$, then the conventional maximum–likelihood estimate is used in the EDA. Otherwise, the estimate based on adaptive variance scaling is used. Note that in the case of maximization we should test for $\theta^{\text{corr}} \geq r$ instead. An experimental illustration of using adaptive variance scaling only when the correlation test was not passed is presented in Figure 9.4. Indeed, adaptive variance scaling is now not always used, preventing the variance from becoming unnecessarily large and speeding up convergence.

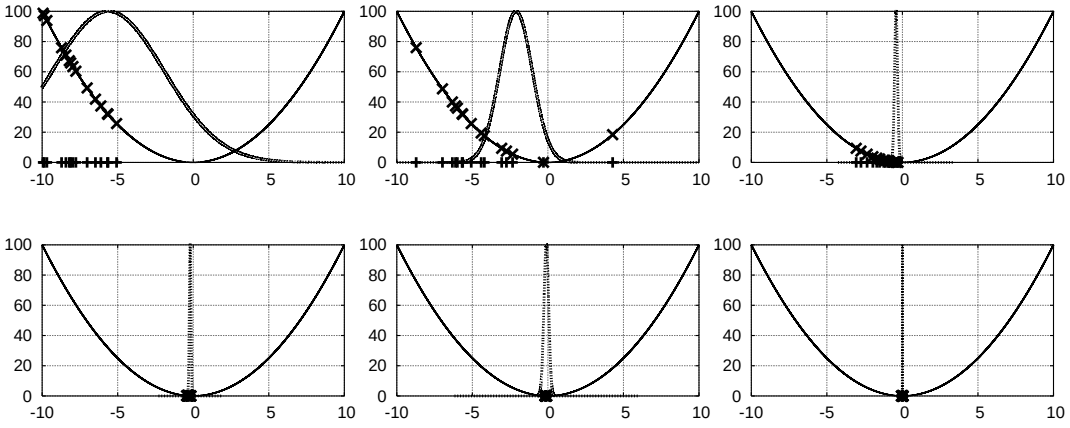


Figure 9.4.: Population and estimated probability distribution (rescaled to fitness range for visualization) in the correlation–triggered adaptive–variance–scaling maximum–likelihood normal EDA in generations 0, 1, 2, 4, 6 and 8 (top–left to bottom–right).

The principle of correlation-triggered adaptive variance scaling is EDA-independent. We integrated it into a continuous EDA based on Bayesian factorizations of normal pdfs, the iterated density- estimation evolutionary algorithm (IDEA, Bosman and Thierens (2000)). The resulting algorithm is called correlation-triggered adaptive variance scaling IDEA (CT-AVS-IDEA). Pseudo-code for CT-AVS-IDEA is presented in Figure 9.5.

Attempting to separate the EDA run into generations where variance scaling appears to be necessary and into phases where it appears not to be necessary directly relates to the notion of diversification and intensification. Scaling the variance beyond its ML-estimate clearly corresponds to diversification of the search. A larger variance increases the area in that solutions are sampled. Once a correlation test as described above dismisses the need to scale the variances beyond the ML-estimates, the EDA can intensify its search on the current center of at-

CT-AVS-IDEA(τ, n, η^{DEC}, $c^{\text{AVS-MAX}}$, θ^{corr})

1. Set generation counter $t = 0$.
2. Initialize population $\mathcal{P}$ with n random individuals.
3. Assign $c^{\text{AVS-MIN}} = 1/c^{\text{AVS-MAX}}$.
4. Assign $\eta^{\text{INC}} = 1/\eta^{\text{DEC}}$.
5. Assign $c^{\text{AVS}} = 1$.
6. Evaluate solutions in $\mathcal{P}$.
7. Store best fitness found in $\mathcal{P}$ in b^t
8. Select best $\lfloor \tau n \rfloor$ individuals and store them in $\mathcal{S}$.
9. If $b^t = b^{t-1}$ then
   (a) assign $c^{\text{AVS}} = c^{\text{AVS}} \cdot \eta^{\text{DEC}}$.
   else
   (b) assign $c^{\text{AVS}} = c^{\text{AVS}} \cdot \eta^{\text{INC}}$.
10. If $c^{\text{AVS}} < c^{\text{AVS-MIN}}$ or $c^{\text{AVS}} > c^{\text{AVS-MAX}}$ then
    assign $c^{\text{AVS}} = c^{\text{AVS-MAX}}$.
11. Estimate Bayesian factorization of normal pdf from $\mathcal{S}$.
12. Compute ranked correlation coefficient r.
13. If $r > \theta^{corr}$ then
    Assign $\mathbf{\Sigma} = c^{\text{AVS}}\mathbf{\Sigma}$.
14. Sample $n - \lfloor \tau n \rfloor$ new candidate solutions from estimated normal pdf (with possibly scaled covariance matrix) and store new candidate solutions in $\mathcal{O}$.
15. Replace worst $n - \lfloor \tau n \rfloor$ individuals in $\mathcal{P}$ with $\mathcal{O}$.
16. Update generation counter, i.e. assign $t = t + 1$.
17. If termination criterion is not met, go back to 6.

Figure 9.5.: CT-AVS-IDEA pseudo-code (minimization).

traction. It has centered around a (possibly local) optimum and reduces the size of the investigated area. In this respect, correlation-triggered variance scaling separates phases 1 and 2 from phase 3 of Chapters 6-8 on the fly.

9.4. Experimental section

9.4.1. Experimental setup

We perform experiments on test functions listed in Table 9.1 using CT-AVS-IDEA, the IDEA without adaptive variance scaling and the CMA-ES Hansen et al. (2003), the current state-of-the-art in evolutionary non-linear optimization.

All functions are unimodal. The optimum for functions 1-7 is obtained by setting $x_i = 0$ for all i. For function 8 the optimum is obtained by setting $x_i = 1$ for all i. The optimum for functions 9 and 10 is obtained by setting $x_i = 0$ for all $i > 1$ and letting x_1 go to ∞. The initialization range used for all functions is $[-10, 5]$, i.e. asymmetric around the optimum and for functions 9 and 10 far away from the optimum for variable x_1.

Name	**Definition**	**Value to reach**
Sphere	$\sum_{i=1}^{l} x_i^2$	10^{-10}
Ellipsoid	$\sum_{i=1}^{l} 10^{6\frac{i-1}{l-1}} x_i^2$	10^{-10}
Cigar	$x_i^2 + \sum_{i=2}^{l} 10^6 x_i^2$	10^{-10}
Tablet	$10^6 x_1^2 + \sum_{i=2}^{l} x_i^2$	10^{-10}
Cigar Tablet	$x_1^2 + \sum_{i=2}^{l-1} 10^4 x_i^2 + 10^8 x_l^2$	10^{-10}
Two Axes	$\sum_{i=1}^{\lfloor l/2 \rfloor} 10^6 x_i^2 + \sum_{i=\lfloor l/2 \rfloor}^{n} x_i^2$	10^{-10}
Different Powers	$\sum_{i=1}^{l} \lvert x_i \rvert^{2+10\frac{i-1}{l-i}}$	10^{-15}
Rosenbrock	$\sum_{i=1}^{l-1} (100 \cdot (x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	10^{-10}
Parabolic Ridge	$-x_1 + 100 \sum_{i=2}^{l} x_i^2$	-10^{-10}
Sharp Ridge	$-x_1 + 100 \sqrt{\sum_{i=2}^{l} x_i^2}$	-10^{-10}

Table 9.1.: Test functions and values to reach.

Using a scalability analysis, the running time complexity of the algorithms is experimentally approximated. To be more specific, it is assessed how the total number of fitness evaluations e and the population size n required to solve the problems to optimality grows with the size of the problem l. Therefore, the dimensionality l was varied: $l \in \{2, 4, 8, 10, 20, 40, 80\}$. For each dimensionality we used a bisection method to obtain the minimally required population size for which the problem's value to reach was found in at least 95 out of 100 independent consecutive runs. The scalability analysis is important, as it allows us to predict whether CT-AVS-IDEA is a tractable approach for solving real-world problems that are often of much higher dimensionality.

For CT-AVS-IDEA we used $\eta^{\text{DEC}} = 0.9$ after pre-experimental testing, i.e. a small multiplication factor to allow for smooth adaptation of the variance multiplication factor. The correlation trigger threshold θ^{corr} was set to $\theta^{\text{corr}} = -0.55$ (see Section 9.4.2). The magnitude of c^{AVS} was bounded from above by $c^{\text{AVS-MAX}} = 10.0$. Following the rule of thumb from Mühlenbein and Mahnig (1999), the selection

threshold τ was set to $\tau = 0.3$ for both CT-AVS-IDEA and the IDEA without variance adaptation.

9.4.2. Setting the correlation trigger threshold

In order to obtain a reasonable value for θ^{corr}, we tested when the ranked correlation coefficient between fitness and density actually triggers scaling of the variance on the sphere function. The sphere function is a single peak and can be solved by EDA without variance scaling. We varied θ^{corr} from -1.0 to 1.0 in steps of 0.01. For each value of θ^{corr}, 100 independent runs of CT-AVS-IDEA on the sphere function in dimensionalities $l \in \{2, 4, 8, 10, 20, 40, 80\}$ were performed. Initial populations were drawn symmetrically around the optimal solution of 0 for all dimensions in a range of $[-7.5, 7.5]$. The population size that was used for a dimensionality l was equal to the minimally required population size for the IDEA to solve this problem optimally. In that case variance scaling is not required because the induced bias of the normal pdf itself suffices to locate the optimum.

Figure 9.6 illustrates the percentage of generations in which variance scaling was nonetheless triggered (averaged over 100 runs). As a rule of thumb, we propose to set θ^{corr} to $\theta^{\text{corr}} = -0.55$. For this value, the number of unnecessary correlation triggers is rather constant and at most 25%. If a smaller value (i.e. closer to -1.0) is chosen, it can be seen from Figure 9.6 that the number of unnecessary correlation triggers will grow with increasing dimensionality. Although the value of -0.55 is rather robust, i.e. values between -0.6 and -0.4 lead to good results, the value for the correlation trigger should not become much larger. If a larger value (i.e. closer to 1.0) is chosen, the scaling of variances was observed from initial experimentation not to be triggered when it is required on slopes.

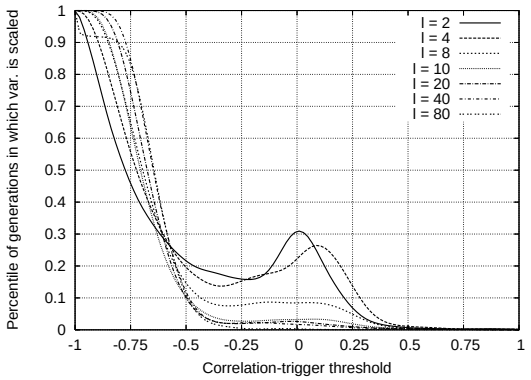


Figure 9.6.: Correlation trigger thresholds.

9.4.3. Results and interpretation

AVS-IDEA, IDEA and CMA-ES

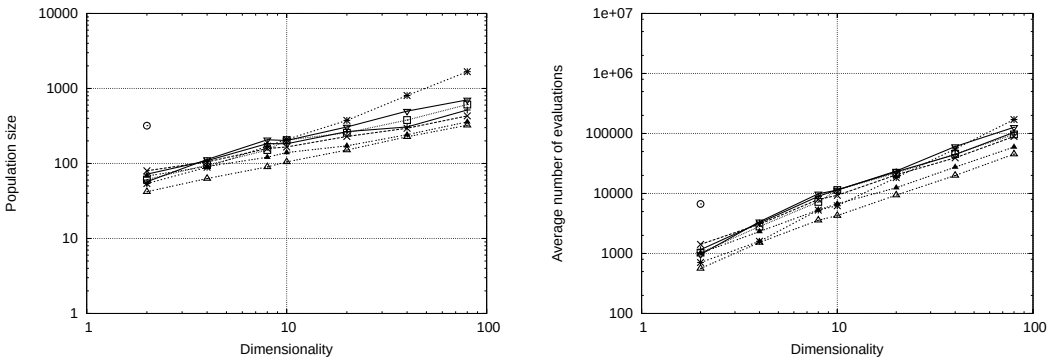
Plots that reveal the influence of problem dimensionality on the average number of evaluations and the minimal population size required to solve the problems are presented in Figure 9.7. As the plots have a log-log scale, straight lines indicate polynomial scalability. Additionally, Table 9.2 shows results from two linear least squares regressions on log-log-scaled data where the average number of evaluations e and the minimally required population size n depend on the dimensionality l of the problems as follows:

$$\log n = \log l^{\alpha} + \epsilon \quad \text{and} \quad \log e = \log l^{\beta} + \epsilon, \tag{9.1}$$

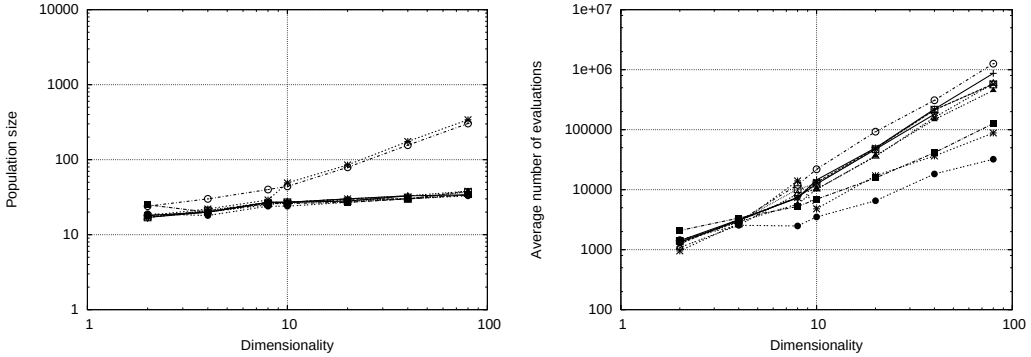
where ϵ is a standard-normally distributed error term.

Results for α indicate that the population size n scales sub-linearly with the problem size l for all regarded algorithms. For the IDEA without covariance adaptation, the population size n grows approximately with the square root of the dimensionality. For AVS-IDEA, the population size n grows even slower. For CMA-ES (Hansen and Ostermeier (2001)) , the population size needs not to be enlarged beyond the initial setting of $\mu = 2$ and $\lambda = 4$ for most functions, except for Rosenbrocks function and the Sharp Ridge function. The reason for this is that in the CMA-ES, the probability distribution used to guide the search is not entirely rebuilt from scratch using only the data in the current set of selected solutions. Instead, the distribution is weighted over a path of generations past and hence represents an accumulation of information.

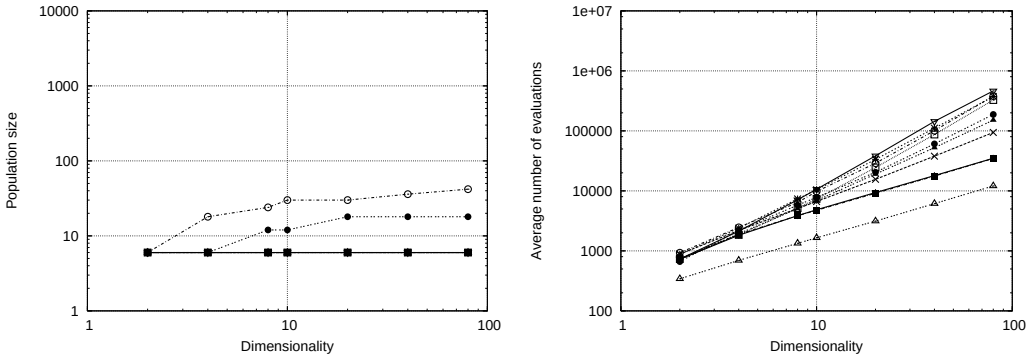
Results for β indicate that the average number of evaluations e for success grows sub-quadratically with l for all regarded algorithms. The average number of evaluations grows faster for the AVS-IDEA than for the IDEA without variance adaptation. However, AVS-IDEA is capable of solving *all* problems in high dimensionality which the IDEA without variance adaptation can not. The IDEA without variance adaptation is incapable of solving Rosenbrocks function, the Parabolic Ridge function and the Sharp Ridge function in higher dimensions. The reason for this is that to find the optimum for the latter two functions, the value for the first variable needs to be moved extremely far outside its initial range. Although the gradient along that direction is straightforward, i.e. it is a simple linear slope, the variance in the IDEA without variance adaptation shrinks too fast and the slope cannot be traveled. In Rosenbrocks function, again the variance shrinks too fast. Even though the optimum lies inside the initial range, the valley in which the optimum is contained is so narrow that the distribution quickly converges to a part inside the valley that is far from the optimum. The bottom of the valley, a curved slope, needs to be traveled to find the optimum. This slope cannot be traveled by the IDEA without variance adaptation.



(a) Normal IDEA without variance adaptation



(b) AVS-IDEA



(c) CMA-ES

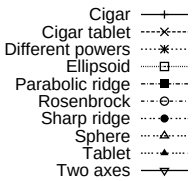


Figure 9.7.: Scalability results for IDEA, AVS IDEA and CMA-ES.

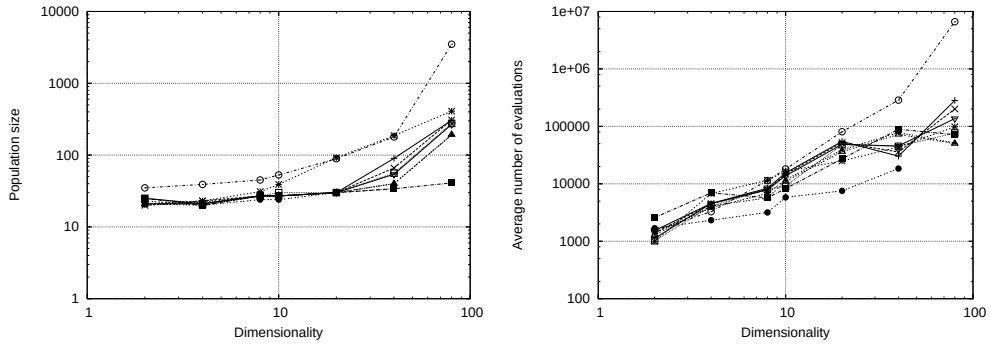


Figure 9.8.: Scalability results for CT-AVS-IDEA (for legend, see Figure 9.7).

Although the CMA-ES has a marginally better scalability than AVS-IDEA on the first half of the benchmark problems, this is not the case for all problems. Moreover, both algorithms scale sub–quadratically in the number of required evaluations to find the optimum.

CT-AVS-IDEA

In Figure 9.8 scalability results are shown for the CT-AVS-IDEA. From the results it can be seen that the addition of the correlation trigger indeed reduces the search effort of the AVS-IDEA. Up to 20 dimensions, although on the one hand the population size scales similarly to the AVS-IDEA, the number of evaluations scale more like those of the normal IDEA, indicating that less evaluations are required because variance scaling is not always required and is consequently correctly detected and signaled by the correlation trigger. However, for a dimensionality of 40 and 80, the correlation trigger reduces in efficiency. On the Sharp Ridge function, the correlation trigger even fails to trigger the scaling of variances altogether. The reason for this is that the correlation trigger and the scaling of variances is done globally for *all* directions, i.e. the entire covariance matrix. For the Sharp Ridge function, all dimensions except one do not require the scaling of variances. The signal obtained in the single non–correlated dimension becomes insignificant as the dimensionality increases and hence variance scaling is no longer triggered. Without variance scaling, the normal IDEA cannot solve the problem and hence, the CT-AVS-IDEA fails. The same will happen for the Parabolic Ridge function, albeit for even higher dimensions and similarly for Rosenbrocks function. For Rosenbrocks function, the problem can still be solved for $l = 80$, albeit clearly no longer in a polynomially scaling fashion, i.e. for even larger dimensionalities the CT-AVS-IDEA will start to behave more like the normal IDEA and hence fail. A solution to this problem may be to factorize the correlation trigger and the scaling of variances. In other words, to have various different variance scaling and correlation triggering mechanisms that are specialized in different directions. A variance-scaling trigger that achieves these goals is proposed in Bosman et al. (2007a).

9.5. Summary and conclusion

In this chapter, we have analyzed the design of continuous EDA starting from the lessons learned from research into discrete EDA and from Chapters 7 and 8. The major lesson learned that the inductive bias of the probabilistic model has to suit to the structure of the optimization problem at hand is valid in any domain. Problem decomposition allows to find efficient descriptions of the problem structure. In the discrete domain, this automatically maps to detecting dependency relations when estimating probability distributions as the basis of the inductive search bias of the optimization algorithm. This understanding then maps to typical GA concepts such as building blocks. In the continuous domain these concepts have no direct equivalent. Hence, before transferring the lessons learned from the discrete domain to the continuous domain, we must first generalize the lessons learned before we specialize them again for the domain at

hand.

In continuous problems, the structure of a problem is characterized by the contours of the function to be optimized. Since the contours can take any shape and hence fitting the problem structure in the continuous case would require the intractable property of universal approximation, it is much more convenient to view problem structure as an arrangement of slopes and peaks in the search space. These simpler substructures are much easier to take into account and to build inductive search biases for.

Continuous EDA rely on normal distributions. We have shown that the bias of the normal distribution does not fit well to slopes due to lack of generalization and as a consequence the EDA can get stuck. When searching around peaks, the bias of the normal distribution suffices to find the optimum. We argue that substructure identification is possible and beneficial in continuous EDA. Substructure identification relates to analyzing the local structure of the current search area so as to find out which shape dominates this search area. To accomplish proper substructure identification, we use the ranked correlation coefficient between the density of the approximated probability distribution and the fitness of the set of selected solutions. On slopes, we then scale the variance of the EDA beyond its maximum–likelihood estimate.

AVS-IDEA was shown to be effective on a test bed of unimodal test functions. In comparison to the IDEA without variance adaptation, it solves all functions from the test bed and requires smaller populations. The total number of fitness evaluations grows faster for AVS-IDEA than for IDEA without variance adaptation. However, for both algorithms the average overall fitness evaluations still grows sub-quadratically with the number of dimensions. Adding the correlation trigger is effective for smaller problems. It does not always work well if the problem dimensionality is higher than 40.

It is an important goal of GEC research to enhance EA such that they are able to solve an increasing array of problems. In this light, we have extended the class of problems that can be solved efficiently and reliably by continuous EDA based on the normal pdf.

Function	**Algorithm**	α	β
Sphere	IDEA	0.5541	1.1635
	AVS-IDEA	0.1994	1.6563
	CMA-ES	0.0000	0.9601
	CT-AVS-IDEA	0.5041	1.1023
Ellipsoid	IDEA	0.6119	1.2171
	AVS-IDEA	0.1870	1.6870
	CT-AVS-IDEA	0.5725	1.1701
	CMA-ES	0.0000	1.5183
Cigar	IDEA	0.5052	1.1865
	AVS-IDEA	0.2125	1.6976
	CT-AVS-IDEA	0.6400	1.2377
	CMA-ES	0.0000	1.1093
Tablet	IDEA	0.4398	1.0860
	AVS-IDEA	0.2066	1.6397
	CT-AVS-IDEA	0.4493	1.1030
	CMA-ES	0.0000	1.4178
Cigar Tablet	IDEA	0.4521	1.1142
	AVS-IDEA	0.1879	1.7155
	CT-AVs-IDEA	0.6192	1.1732
	CMA-ES	0.0000	1.2431
Two Axes	IDEA	0.6603	1.2854
	AVS-IDEA	0.2177	1.6551
	CT-AVS-IDEA	0.5879	1.1530
	CMA-ES	0.0000	1.7208
Different Powers	IDEA	0.9355	1.4983
	AVS-IDEA	0.8419	1.1692
	CT-AVS-IDEA	0.8568	1.1016
	CMA-ES	0.0000	1.5845
Rosenbrock	IDEA	not solved	
	AVS-IDEA	0.7475	1.9154
	CT-AVS-IDEA	0.8830	1.9937
	CMA-ES	0.6885	1.4872
Parabolic Ridge	IDEA	not solved	
	AVS-IDEA	0.1064	1.1160
	CT-AVS-IDEA	0.1686	1.0536
	CMA-ES	0.0000	1.0853
Sharp Ridge	IDEA	not solved	
	AVS-IDEA	0.1678	0.8563
	CT-AVS-IDEA	0.1570	0.7913
	CMA-ES	0.5228	1.4764

Table 9.2.: Regression coefficients for scalability.

10. CT-AVS-IDEA solves stochastic transportation problems

10.1. Introduction

As shown in Section 9.4, adaptive variance scaling significantly improves the performance of continuous EDA. CT-AVS-IDEA solves complicated nonlinear benchmark problems reliably and efficiently. This chapter assesses the performance of CT-AVS-IDEA beyond artificial benchmark functions and applies it to the stochastic transportation problem (STP).

The stochastic transportation problem arises in supply chain management when coordinated decisions on shipment and order quantities must be made under uncertainty. Material flows from sources where inventory and production capacity is available to sinks representing customers or markets whose stochastic demand has to be satisfied. Sources and sinks are linked through transportation possibilities. Generally, demand uncertainty can be handled in a proactive or reactive manner, see Kenyon and Morton (2002). A proactive approach optimizes total expected costs of stock-outs and leftover inventory. This basic trade-off is analyzed in the one-period one-product inventory control problem, often referred to as the newsvendor-problem, for a review see Khouja (1999).

Stochastic transportation problems are important in practice. They can readily be extended towards more complex stochastic and dynamic settings, see Arnold (1987) for an overview. It arises in strategic network design and operational transportation planning of advanced planning systems, see Fleischmann (2005). The STP is a sub-problem of strategic network design when locations are fixed. On an operational level, the STP provides inventory levels that should be held in short-term at supplying sources. Allen (1958) studies the related problem of redistributing stock using transshipments. In this respect, a classification of nodes in a supply network into sources and sinks is part of the decisions. The nodes hold initial inventory that is redistributed before demand realizes.

One important means to distinguish between stochastic optimization problems is the sequence and timing of decisions and realizations of random variables. If customer/market demand is known before the shipment quantities must be set, the problem reduces to a deterministic transportation problem, see Domschke (2007). In this situation, the largest possible advantage of postponing the transportation decision is realized. The problem can be formulated as a stochastic program

if the stock levels and shipment quantities must be set before demand realizes. Chance-constraint approaches are required, if the expected degree and/or impact of stock outs is bounded by a service level agreement, see Witten and Zimmermann (1978).

A different approach is taken, if decisions related to compensation are modeled, e.g., emergency transshipments in case of stock outs and discounted sales in case of leftover inventory. Thereby, a first-stage decision sets initial inventory levels, capacities and/or the network structure while anticipating the expected cost that is caused by compensation actions at a second stage. This approach is considered in this chapter. The first stage decisions comprise the choice of sources and material supply for a transportation problem. The second stage describes cost consequences of leftover inventory or stock-outs in a one-period inventory model.

The STP that results from combining the classical transportation problem with the one-period newsvendor-problem is convex under linear constraints and in principle solvable by methods of convex optimization. However, an integration of concave economies of scale in production and of expected convex mismatch costs at the second stage yields a multimodal problem that is intractable for standard non-linear programming approaches.

Meta-heuristics are routinely used to solve combinatorial optimization problems in logistics. On the contrary, they are still rarely used to solve nonlinear continuous optimization problems. The goal of this chapter is to solve the STP with the CT-AVS-IDEA, see Chapter 9.3.2. This will provide insights into the performance of CT-AVS-IDEA beyond artificial test problems.

Section 10.2 presents the classical STP and reviews solution methods that have been proposed in the literature. The impact of integrating economies of scale on the problem complexity is discussed afterwards. Section 10.3 reports on results from an experimental study that is conducted using CT-AVS-IDEA on a test bed of STPs. Linearly approximated programs are used as benchmarks. The chapter is concluded in Section 10.4. It has been published in Grahl and Minner (2006).

10.2. Stochastic transportation problems

10.2.1. The classical stochastic transportation problem

Consider m supplying nodes (sources) $i = 1, 2, \ldots, m$ that offer a_i units of capacity per period of time. n demand nodes (sinks) $j = 1, 2, \ldots, n$ face stochastic demand D_j. The distribution functions ϕ_j and the cumulative distribution functions F_j are known. As a simplification, it is assumed that demand is uncorrelated between customers. The goal is to decide on stock levels y_j such that total cost

are minimized. Unmet demand causes opportunity costs of lost sales (underage costs) of p_j per unit at sink j. Leftover inventories cost h_j per unit (overage costs). The expected lost sales and inventory costs at sink j with stock level y_j are

$$f_j(y_j) = -cy_j + h_j \int_0^{y_j} (y_j - x)\phi_j(x)dx + p_j \int_{y_j}^{\infty} (x - y_j)\phi_j(x)dx. \tag{10.1}$$

The optimal solution for y_j that minimizes (10.1) results from the well known critical ratio

$$F_j(y_j) = \frac{p_j - c}{p_j + h_j}. \tag{10.2}$$

Setting an optimal value for y_i results in a probability $F_j(y_j)$ which equals the ratio between underage costs and the sum of underage and overage costs.

Producing a single unit at source i and transporting it to sink j causes variable linear costs of c_{ij}. The goal is to minimize expected total costs per period that consist of production, transportation, inventory holding and opportunity costs of lost sales. Apart from setting the stock levels y_j, shipment volumes x_{ij} between sources and sinks are set in equalities (10.4) such that y_j equals the total amount that it receives. Inequalities (10.5) enforce capacity limitations at the sources. The resulting non-linear optimization problem with linear constraints is

$$\min Z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} f_j(y_j) \tag{10.3}$$

$$s.t. \sum_{i=1}^{m} x_{ij} = y_j \qquad \forall j = 1, 2, \ldots, n \tag{10.4}$$

$$\sum_{j=1}^{n} x_{ij} \leq a_i \qquad \forall i = 1, 2, \ldots, m \tag{10.5}$$

$$x_{ij} \geq 0 \qquad \forall i = 1, 2, \ldots, m; j = 1, 2 \ldots, n \tag{10.6}$$

$$y_j \geq 0 \qquad \forall j = 1, 2, \ldots, n. \tag{10.7}$$

The above model comprises two special cases. It reduces to the classical transportation problem if $y_j = b_j, b_j \in \mathbb{R}_+^0$. If an allocation of sources to sinks is fixed, the capacity constraint is not binding and each sink is supplied from a single source only, the problem decomposes into independent newsvendor problems.

Exact optimization algorithms are most prominent in the literature. Elmaghraby (1960) presents a method to systematically evaluate Kuhn-Tucker conditions. Qi (1985) improves on this work by considering only a single dimension per iteration. The convex minimization problem under linear constraints is solvable with methods of convex optimization, see Horst and Tuy (1998). Williams (1963) uses an approach dating back to Dantzig. Cooper and LeBlanc (1977) and LeBlanc et al. (1985) use the Frank-Wolfe approach. Holmberg and Jörnsten (1984) discuss pricing-based Dantzig-Wolfe and resource-based Benders decomposition methods and develop a combined algorithm that relaxes the complicated master problem. Holmberg (1995) compares decomposition and linearization approaches for the STP and reports on advantages from using decomposition approaches.

Heuristic approaches result from stopping exact methods before they reach the optimal solution. Wilson (1975) presents a problem-specific heuristic that employs cost approximations.

10.2.2. Integrating economies of scale

We extend the basic model from Section 10.2.1 by assuming economies of scale of production. Production costs at source i follow a non-linear and concave cost function $g_i(z_i)$, where z_i is the total production quantity and capacity at source i.

$$\min Z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} g_i(z_i) + \sum_{j=1}^{n} f_j(y_j) \tag{10.8}$$

$$s.t. \sum_{i=1}^{m} x_{ij} = y_j \qquad \forall j = 1,2,\ldots,n \tag{10.9}$$

$$\sum_{j=1}^{n} x_{ij} = z_i \qquad \forall i = 1,2,\ldots,m \tag{10.10}$$

$$0 \le z_i \le a_i \qquad \forall i = 1,2,\ldots,m \tag{10.11}$$

$$x_{ij} \ge 0 \qquad \forall i = 1,2,\ldots,m; j = 1,2,\ldots,n \tag{10.12}$$

$$y_j \ge 0 \qquad \forall j = 1,2,\ldots,n \tag{10.13}$$

$$z_i \ge 0 \qquad \forall i = 1,2,\ldots,m \tag{10.14}$$

The above formulation of $g_i(z_i)$ includes the special case that production costs contain a fixed component that arises only if $z_i > 0$, as assumed in warehouse location problems, see Drezner and Hamacher (2002) for a review. Constraints

(10.9) and (10.10) link produced, shipped and stored goods, while (10.11) ensures that the capacity constraint is met.

The objective function consists of a concave part that describes production costs, a linear part that describes transportation costs, and a convex part that describes inventory holding and lost sales costs. It is thus neither concave nor convex and can have multiple local optima. Non-convex optimization problems under linear constraints are in general NP-hard, see Pardalos and Rosen (1984). Holmberg and Tuy (1999) develop a branch and bound algorithm to solve the above STP.

10.3. Experimental section

10.3.1. Experimental design

We use CT-AVS-IDEA (see Section 9.3.2) to solve the STP described in Section 10.2.2. The number of sources is varied in $m \in \{2, 5, 10\}$, the number of sinks is varied in $n \in \{10, 35, 50\}$ resulting in 9 network configurations. $K = 50$ random instances are generated for each configuration as follows. The sources and sinks are randomly placed in a square area that is 100×100 units large. Variable transportation costs c_{ij} from source i to sink j are the euclidean distance scaled by a random factor $UF \sim U[1; 1.5]$. The production cost function $g_i(z_i)$

$$g_i(z_i) = \beta_i z_i^{\alpha_i} \tag{10.15}$$

is concave. The parameter $\alpha_i \sim U[0.5; 1]$, and $\beta_i \sim U[100; 200]$. Demand D_j is normally distributed with mean $\mu_j \sim U[40; 200]$ and standard deviation $\sigma_j = \mu_j \epsilon_j$. The coefficient of variation $\epsilon_j \sim U[0.1; 0.5]$. Underage costs p_j are uniformly distributed in $[100; 200]$, overage costs h_j are uniformly distributed in $[3; 6]$. Capacity constraints are neglected.

For the special case of normally distributed demand, the expected lost sales and inventory holding costs are given as

$$f_j(y_j) = p_j(\mu_j - y_j) + (h_j + p_j)(y_j - \mu_j)\Phi_{0,1}\left(\frac{y_j - \mu_j}{\sigma_j}\right) + (h_j + p_j)\sigma_j \phi_{0,1}\left(\frac{y_j - \mu_j}{\sigma_j}\right), \tag{10.16}$$

where $\Phi_{0,1}$ is the cumulated density function of the standard normal distribution, and $\phi_{0,1}$ is the density function of the standard normal distribution.

Benchmarks were obtained by piece-wise linearization of production and mismatch cost functions. The resulting mixed-integer linear program was solved by XPress-MP. Production costs $g_i(z_i)$ are concave in z_i, mismatch costs $f_j(y_j)$ are convex in y_j. In order to obtain a lower bound Z_l for the optimal costs, $g_i(Z_i)$ was linearized using secants, and $f_j(y_j)$ was linearized using tangents. An upper

bound Z_u on the costs was obtained conversely. Z_u and Z_l converge when the number of breakpoints in the piece-wise defined functions increases.

The first break-point for lingering $f_j(y_j)$ is $y_j = 0\ \forall j = 1, 2, \ldots, n$. An upper bound y^{MAX} resulted from solving (10.2) using the minimal transportation costs $c_{ij}^{MIN} = \min\{c_{ij} | i = 1, 2, \ldots, m; j = 1, 2, \ldots, n\}$ as per unit price for sink j.

The first break-point for linearizing $g_i(z_i)$ is $z_i = 0 \forall i = 1, 2, \ldots, m$. An upper bound z_i^{MAX} results from choosing the largest possible sensible amount

$$z_i^{MAX} = \sum_{j=1}^{n} y_j^{MAX} \forall i = 1, 2, \ldots, m. \tag{10.17}$$

The cumulated distribution function of the normal density was approximated by the polynomial approximation 26.2.19, the quantile function of the normal cumulative distribution function was approximated by the rational approximation 26.2.22 from Abramowitz and Stegun (1972).

The population size used by CT-AVS-IDEA is set to $2nm$, the ratio of selected individuals τ is set to $\tau = 0.3$. θ^{CORR} is set to 0.55. A smooth adaptation of the variance is obtained by setting η^{DEC} to 0.9. Furthermore, $c^{\text{AVS-MAX}} = 10.0$. CT-AVS-IDEA was terminated as soon as the cost difference between the best and the worst solution in a population was smaller than 1.0.

10.3.2. Results and interpretation

A lower bound Z_l^k was derived for each of the $k = 1, 2, \ldots, 50$ instances using successive linear approximation and XPress-MP. The number of breakpoints used in the linearization was chosen such that the lower bounds were not further apart than 2-3% of total cost. It is used as a benchmark and is compared to the best found solution x_b^k of CT-AVS-IDEA. A cost increase Δ^k is defined as

$$\Delta^k = \frac{Z(x_b^k) - Z_l^k}{Z_l^k} \cdot 100\%. \tag{10.18}$$

The results are listed in Table 10.1. Minimal, average and maximal cost increases that are obtained from CT-AVS-IDEA over the bound are reported.

It is immediately apparent from the results that CT-AVS-IDEA is capable of finding optimal or near-optimal solutions of the STP without exploiting any problem-specific knowledge besides the objective function.

n	10			35			50		
m	min	avg	max	min	avg	max	min	avg	max
2	0.01	0.04	0.38	0.00	0.02	0.61	0.01	0.17	0.71
5	0.00	0.21	3.16	0.02	0.20	0.84	0.05	0.43	1.76
10	0.00	0.42	2.60	0.78	1.82	3.76	0.52	2.50	8.80

Table 10.1.: Cost penalty of using CT-AVS-IDEA over lower bound in percent.

10.4. Summary and conclusion

The mainstream approach to solving the STP is hand-tailoring problem specific solution methods for its variants. Specialized methods exist for linear, concave, or convex cost structures. We have shown, that CT-AVS-IDEA can solve the STP. This is an important result, as it demonstrates the applicability of CT-AVS-IDEA beyond artificial benchmark problems. From an applied point of view, the availability of robust BBO algorithms such as CT-AVS-IDEA opens the road for resolving difficult planning problems at minimal design efforts without having to face the downsides of classical techniques. This advantage gets more pronounced, if non-linear concave transportation costs (see Fleischmann (1993) are used or customer demand is correlated between markets, rendering a linearization of the objective function more complex and time-consuming.

11. Summary, conclusion and outlook

11.1. Summary

This thesis centered around the following two research topics. In its first part (Chapters 3-5), existing EDA theory and discrete EDA were applied to problems in logistics in order to bridge the gap between discrete EDA theory and application. In its second part (Chapters 6-10) continuous EDA were analyzed, redesigned, implemented and tested.

The second chapter introduced genetic algorithms, estimation of distribution algorithms and basic theoretical concepts like factorized search distributions and the notion of a problem decomposition.

The first part of the thesis started in Chapter 3, where EDA theory was applied to the warehouse location problem. A subtle numbering defect was discovered. After explaining the defect formally, an experimental comparison of sGA and the BOA was carried out on test instances that are widely used in the literature. For the instances considered, EDA are found to be more efficient than sGA. Chapter 4 applied the sGA, the BOA and the (1+1)-EA on safety stock allocation problems in serial, divergent, convergent and general topologies. An experimental study was conducted, on basis of which topology-algorithm matchings were proposed. The sole adaptation made to the EA was the integration of the fitness function on basis of a fixed-length binary encoding. The (1+1)-EA was found reliable for solving serial problems. For other network topologies, the BOA dominated the other EA in terms of reliability or runtime.

In Chapter 5, the single-product dynamic demand lot-sizing problem and the dynamic joint-replenishment problem were analyzed and it was found that the problems are decomposable in the sense of EDA theory. Furthermore, a state-of-the-art EDA was used to solve instances of both problems in a scalability analysis. The results were in accordance with existing scalability theory of EDA as the run-time of the EDA measured in number of fitness evaluations grew with a low-order polynom depending on the size of the instances.

Chapter 6 introduced the second part of the thesis by setting the stage for the formal analysis of continuous EDA. It therefore decomposed overall convergence into three simpler to understand phases that were assumed independent. The first phase represents a situation in which a continuous EDA is searching far from the optimum. Population statistics for this phase were derived analytically

in Chapter 7. Phase two generalizes to a situation in which the algorithm has moved towards the optimum and a significant proportion of solutions is optimal. It was shown in Section 8.1 that a preferred sampling variance exists for a simple setting that should be chosen in order to maximize the ratio of optimal solutions. Once the mean is located on the optimal solution and the variance must shrink to sample optimal solutions reliably, the EDA has reached phase three. A runtime result for this phase was derived in Section 8.2.

The results indicated that a proper tuning of variances is essential for the success of continuous EDA that are based on normal distributions. To be more precise, using the well-known maximum likelihood estimators for the normal distribution does not automatically enable efficient search, even if a full covariance matrix is estimated. In this respect, continuous EDA differ substantially from their discrete counterpart, where maximum likelihood estimation of model parameters is sufficient, given that the structure of the model can capture the decomposition of the problem.

Chapter 9 discussed in detail potential pitfalls in the adaptation of discrete EDA to the continuous domain. Additionally, it proposes a simple remedy to enhance the performance of continuous EDA: correlation-triggered adaptive variance scaling. This concept was integrated into the IDEA algorithm in Section 9.3.2. The resulting algorithm CT-AVS-IDEA was tested on artificial non-linear test functions in Section 9.4 and was found to solve complicated non-linear problems reliably to global optimality. Furthermore, CT-AVS-IDEA was applied to the stochastic transportation problem in Chapter 10, generating solutions that are close to optimality in reasonable time.

11.2. Conclusion

This section will conclude the major findings of this thesis. First, it summarizes its contribution to the interface between EDA theory and logistics, afterwards its contribution in the field of continuous EDA are discussed.

This thesis attempted to bridge the gap between EDA theory and the application of EDA. Recent advances in evolutionary computation that have, e.g., lead to estimation of distribution algorithms, remain largely unnoticed in practice and applied optimization. Successful applications of EDA are required that illustrate the drawbacks of simple GA and drive the diffusion of the EDA paradigm in practice.

In this respect, EDA theory was used to analyze the structure of the uncapacitated warehouse location problem. It was shown, that the numbering of warehouses can significantly increase or decrease the performance of a simple GA. This is clearly an undesired behavior, especially as it is virtually impossible to

judge the quality of the a chosen numbering a priori. In order to overcome this problem, linkage learning techniques are needed. The BOA (and any other EDA that can capture multivariate interactions) does not suffer from the numbering defect. Additionally, it was found to dominate the simple GA in terms of efficiency and reliability. Since significant speed-ups are obtained from tightly-linked codings, optimization practitioners should investigate possibility to obtain such codings, or use linkage learning techniques when solving location problems.

Safety stock allocation is one field of research where the literature is dominated by problem-specific approaches. The thesis challenged this approach and used evolutionary black-box-optimizers to solve safety stock problems. These algorithms exploited an extreme-point property that allowed the encoding of solutions on a fixed-length binary string. No further modification was made to the built-in search mechanics of the EA. Nonetheless, the algorithms were able to routinely solve safety stock allocation problems in different topologies to global optimality. The BOA was especially successful solving problems in complex structures that arise in the real world. Here, the simple GA turned out to be less efficient and less reliable.

Finally, the structure of single-product and multi-product dynamic demand lot-sizing problems was analyzed. It was shown that these problems are decomposable in the sense of EDA theory. Experimental scalability results highlighted the potential of EDA in inventory management.

Summarizing the findings of the first part of the thesis, important insights in the behavior of search algorithms can be obtained from the application of EDA theory to optimization problems. This was demonstrated for warehouse location problems. Furthermore, evolutionary black box optimization is a flexible and efficient means to solving related variants of non-linear optimization problems that would require hand-tailored methodologies otherwise. This was demonstrated for safety stock allocation problems in different network topologies. Finally, EDA theory has been developed for artificial test problems. Nonetheless, it matches the behavior of EDA on practical problems that are decomposable. This could be shown for dynamic demand lot-sizing problems and the joint replenishment problem.

It must be noted though, that using un-modified versions of state-of-the-art multivariate EDA for very large instances of (combinatorial) optimization problems might take very long due to time demand model building. This potential drawback must be traded off against reliability and ease of use. Efficiency enhancements like sporadic model building (Sastry et al. (2004)) or BBO local search (Lima et al. (2006)) can be integrated into EDA to reduce the time required for model building.

Regarding the second part of the thesis, first continuous EDA were designed as direct counterparts of successful discrete EDA. A variety of probabilistic models based on the normal distribution were used. It was hoped, that an increase in

the modeling capacity of the distribution used should leverage the performance of continuous EDA. This thesis contributed to understanding why this direct adaptation was not successful. It showed that the variances obtained through maximum likelihood estimation decrease exponentially fast. This can be problematic if the algorithm is initialized far from the optimal solution and can easily cause premature convergence. One way to solve this problem would be to use a more involved probability distribution but the Gaussian distribution in order to better model the structure of the problem. However, it was discussed in this thesis that problem structure in continuous spaces is expressed through fitness landscape contours. These contours can be of virtually any shape. Since arbitrary structure approximation in high dimensions is computationally not tractable, a different approach was taken. In order to prevent premature convergence, the Gaussian distribution was kept, but the sampling variance was enlarged beyond its maximum-likelihood estimate. Variance enlargement was triggered only if necessary, that is on slope-like regions of the search space. A trigger based on correlation between density and fitness was used to achieve this goal.

The resulting CT-AVS-IDEA significantly improved upon the state-of-the-art of continuous EDA and is among the most efficient continuous EA available. It increased efficiency on problems that were solvable before, and beyond that enlarged the class of problems that continuous EDA are able to solve reliably. In this light, the thesis demonstrated both formally and experimentally that a proper tuning of the parameters of the distributions used is essential for the success of continuous EDA and exemplified how efficient continuous EDA can be designed.

11.3. Outlook

The uWLP considered in Chapter 3 neglects practical properties like limited warehouse capacities and/or multiple distribution stages. Furthermore, problem-specific meta-heuristics that are tailored towards certain problem classes have been designed, e.g., in Kratica et al. (1996). It is interesting to analyze, how well sGA and EDA are able to solve multilevel location problems and whether the numbering effect plays an important role when solving these problems with EA. Additionally, the integration of local search into baseline algorithms like the sGA as done in Kratica et al. (1996) is likely to influence the strength of the numbering effect. It is tempting to generalize the numbering effect to a problem that arises when geographical information is mapped onto a binary string.

Structural insights that have been gained for the uncapacitated warehouse location problem might be usable to design EDA that are especially efficient for this problem. One attempt is to reduce the complexity of the probabilistic model to interactions that are important.

Regarding the results from Chapter 4, it seems worthwhile to investigate heuristic approaches for large scale safety stock allocation problems. A first approach can be to hybridize global and local search. EDA can function as global searchers while local search (see Minner (2000)) improves upon selected genotypes. This will lead to operators specifically designed for safety stock allocation problems. In order to reduce the runtime of EDA, the search space for model building might be reducible. Therefore, in-depth knowledge about the probabilistic models that are built by EDA is a requirement.

Moreover, the problem difficulty of safety stock allocation problems has not been analyzed theoretically, and it is hard to tell what makes an instance hard for heuristic search algorithms. Possible first steps can be to assess the locality of binary encodings, see Rothlauf (2006) or fitness landscape analysis, see Merz and Freisleben (1999).

Lot-sizing problems typically arise in capacitated multi-level settings where a bill-of-material structure couples material flows. This class of problems has received considerable attention in the last decades and might serve as a next step in the application of EDA in inventory management.

The large population sizes that have been observed in this thesis are undesired in practice because they lead to high computational costs when probabilistic models are built. In order to overcome this drawback, efficiency enhancements have been designed, see Sastry et al. (2004). Specifically, sporadic model building should be addressed, as it provides significant speed-ups at little effort, see Pelikan et al. (2006c).

The theoretical investigations made in Section 8.1 should be generalized to several dimensions and/or other search distributions. These results can act as a benchmark for practical variance scaling policies. Further, it will be interesting to work out variance scaling policies that are capable of emulating optimal variances.

Runtime results such as the one derived in Section 8.2 can be starting points for a systematic comparison of the performance of EDA with other evolutionary and non-evolutionary algorithms.

Bibliography

Abramowitz, M., I. A. Stegun. 1972. *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*. Courier Dover Publications, New York.

Ahn, C. W., R. S. Ramakrishna, D. E. Goldberg. 2004. Real-coded Bayesian optimization algorithm: Bringing the strength of BOA into the continuous world. K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P. L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, A. Tyrrell, eds., *Genetic and Evolutionary Computation – GECCO-2004*, *Lecture Notes in Computer Science*, vol. 3102. Springer, Seattle, Washington, 840–851.

Akinc, U., B. M. Khumawala. 1977. An efficient branch and bound algorithm for the capacitated warehouse location problem. *Management Science* **23** 585–594.

Allen, S. G. 1958. Redistribution of total stock over several user locations. *Naval Research Logistics Quarterly* **5** 337–345.

Anderson, T. W. 2003. *An Introduction to Multivariate Statistical Analysis*. 3rd ed. John Wiley & Sons Inc., New York.

Arkin, E., D. Joneja, R. Roundy. 1989. Computational complexity of uncapacitated multi-echelon production planning problems. *Operations Research Letters* **8** 61–66.

Arnold, K.-P. 1987. *Stochastische Transportprobleme*. 17, Dr. Kovač, Hamburg.

Axsäter, S. 2003. Supply chain operations: Serial and distribution inventory systems. A. G. de Kok, S. C. Graves, eds., *Supply Chain Management: Design, Coordination and Operation*. Handbooks in Operations Research and Management Science, North-Holland, Amsterdam, 525–559.

Baluja, S. 1994. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. CMU-CS-94-163, Carnegie Mellon University.

Baluja, S., S. Davies. 1997. Using optimal dependency–trees for combinatorial optimization: Learning the structure of the search space. D. H. Fisher, ed., *Proceedings of the 1997 International Conference on Machine Learning*. Morgan Kaufmann, Madison, Wisconsin, 30–38.

Barahona, F., F. Chudak. 2005. Near-optimal solutions to large scale facility location problems. *Discrete Optimization* **2** 35–50.

Beasley, J. E. 1988. An algorithm for solving large capacitated warehouse location problems. *European Journal of Operational Research* **33** 314–325.

Beasley, J. E. 1993. Lagrangean heuristics for location problems. *European Journal of Operational Research* **65** 383–399.

Bengoetxea, E., P. Larrañaga, I. Bloch, A. Perchant, C. Boeres. 2002. Learning and simulation of Bayesian networks applied to inexact graph matching. *Pattern Recognition* **35** 2867–2880.

Bilde, O., J. Krarup. 1977. Sharp lower bounds and efficient algorithms for the simple plant location problem. *Annals of Discrete Mathematics* **1** 79–97.

Blanco, R., I. Inza, P. Larrañaga. 2003. Learning Bayesian networks in the space of structures by estimation of distribution algorithms. *International Journal of Intelligent Systems* **18** 205–220.

Blanco, R., P. Larrañaga, I. Inza, B. Sierra. 2001. Selection of highly accurate genes for cancer classification by estimation of distribution algorithms. P. Lucas, ed., *Proceedings of the Bayesian Models in Medicine Workshop at the 8th Artificial Intelligence in Medicine in Europe AIME–2001*. 29–34.

Blickle, T., L. Thiele. 1996. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation* **4** 361–394.

Blum, C. 2004. Theoretical and practical aspects of ant colony optimization. Ph.D. thesis, Université Libre de Bruxelles.

Blum, C., M. Dorigo. 2005. Search bias in ant colony optimization: On the role of competition-balanced systems. *IEEE Transactions on Evolutionary Computation* **9** 159–174.

Boctor, F. F., G. Laporte, J. Renaud. 2004. Models and algorithms for the dynamic joint replenishment problem. *International Journal of Production Research* **42** 2667–2678.

Bosman, P. A. N. 2003. Design and application of iterated density-estimation evolutionary algorithms. Ph.D. thesis, University of Utrecht, Institute of Information and Computer Science.

Bosman, P. A. N., E. D. De Jong. 2004. Learning probabilistic tree grammars for genetic programming. X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. M. Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiño, A. Kabán, H.-P. Schwefel, eds., *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference*, *Lecture Notes in Computer Science*, vol. 3242. Springer, 192–201.

Bosman, P. A. N., J. Grahl. 2008. Matching inductive search bias and problem structure in continuous estimation-of-distribution algorithms. *European Journal of Operational Research* **185** 1246–1264.

Bosman, P. A. N., J. Grahl, F. Rothlauf. 2007a. SDR: A better trigger for adaptive variance scaling in normal EDAs. D. Thierens, H.-G. Beyer, M. Birattari, J. Bongard, J. Branke, J. A. Clark, D. Cliff, C. B. Congdon, K. Deb, B. Doerr, T. Kovacs, S. Kumar, J. F. Miller, J. Moore, F. Neumann, M. Pelikan, R. Poli, Sastry K, K. O. Stanley, T. Stützle, R. A. Watson, I. Wegener, eds., *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM Press, New York, New York, USA, 492–499.

Bosman, P. A. N., J. Grahl, D. Thierens. 2007b. Adapted maximum-likelihood gaussian models for numerical optimization with continuous EDA. Under Review.

Bosman, P. A. N., D. Thierens. 1999. Linkage information processing in distribution estimation algorithms. W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, R. E. Smith, eds., *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*. Morgan Kaufmann, Orlando, Florida, USA, 60–67.

Bosman, P. A. N., D. Thierens. 2000. Expanding From Discrete to Continuous Estimation of Distribution Algorithms: The IDEA. M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo Guervós, H.-P. Schwefel, eds., *Parallel Problem Solving from Nature - PPSN VI, 6th International Conference*, *Lecture Notes in Computer Science*, vol. 1917. Springer, 767–776.

Bosman, P. A. N., D. Thierens. 2001a. Advancing continuous IDEAs with mixture distributions and factorization selection metrics. R. B. Heckendorn, ed., *2001 Genetic and Evolutionary Computation Conference Workshop Program*. San Francisco, California, 208–212.

Bosman, P. A. N., D. Thierens. 2001b. Crossing the road to efficient IDEAs for permutation problems. L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, E. Burke, eds., *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. Morgan Kaufmann, San Francisco, California, 219–226.

Bosman, P. A. N., D. Thierens. 2001c. New IDEAs and more ICE by learning and using unconditional permutation factorizations. E. D. Goodman, ed., *2001 Genetic and Evolutionary Computation Conference Late Breaking Papers*. San Francisco, California, 16–23.

Bosman, P. A. N., D. Thierens. 2002. Multi–objective optimization with diversity preserving mixture–based iterated density estimation evolutionary algorithms. *International Journal of Approximate Reasoning* **31** 259–289.

Bosman, P. A. N., D. Thierens. 2003. The balance between proximity and diversity in multi–objective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* **7** 174–188.

Burnham, K. P., D. R. Anderson. 2002. *Model Selection and Multimodel Inference: A Practical-Theoretic Approach*. Springer, Berlin, Heidelberg, New York.

Cho, D.-Y., B.-T. Zhang. 2001. Continuous estimation of distribution algorithms with probabilistic principal component analysis. *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*. IEEE Press, Seoul, Korea, 521–526.

Cho, D.-Y., B.-T. Zhang. 2002. Evolutionary optimization by distribution estimation with mixtures of factor analyzers. D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, M. Shackleton, eds., *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*. IEEE Press, 1396–1401.

Cho, D.-Y., B.-T. Zhang. 2004. Evolutionary continuous optimization by distribution estimation with variational Bayesian independent component analyzers mixture model. X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. M. Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiño, A. Kabán, H.-P. Schwefel, eds., *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference*, *Lecture Notes in Computer Science*, vol. 3242. Springer, 212–221.

Chow, C. K., C. N. Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory* **14** 462–467.

Cooper, L., L. J. LeBlanc. 1977. Stochastic transportation problems and other network related convex problems. *Naval Research Logistics Quarterly* **24** 327–336.

Cornuéjols, G., G. L. Nemhauser, L. A. Wolsey. 1990. The uncapacitated facility location problem. P. B. Mirchandani, R. L. Francis, eds., *Discrete Location Theory*. Wiley-Interscience, New York, 119–171.

Costa, M., E. Minisci. 2003. MOPED: A multi-objective parzen-based estimation of distribution for continuous problems. C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, L. Thiele, eds., *Evolutionary Multi-Criterion Optimization Second International Conference – EMO 2003*, *Lecture notes in Computer Science*, vol. 2632. Springer, Berlin, Heidelberg, New York, 282–294.

Cotta, C., J. I. van Hemert, eds. 2007. *Evolutionary Computation in Combinatorial Optimization, 7th European Conference*, *Lecture Notes in Computer Science*, vol. 4446. Springer, Berlin, Heidelberg, New York.

Daskin, M. S. 1995. *Network and Discrete Location: Models, Algorithms, and Applications*. Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, New York.

De Bonet, S. J., C. L. Isbell, P. Viola. 1997. MIMIC: Finding optima by estimating probability densities. M. C. Mozer, M. I. Jordan, T. Petsche, eds., *Advances in Neural Information Processing Systems*, vol. 9. The MIT Press, 424.

Deb, K., D. E. Goldberg. 1993. Analysing deception in trap functions. L. D. Whitley, ed., *Foundations of Genetic Algorithms 2*. Morgan Kaufmann, San Francisco, California, 93–108.

DeBodt, M., L. Gelders, L. van Wassenhove. 1984. Lot sizing under dynamic demand conditions: A review. *Engineering Costs and Production Economics* **8** 165–187.

Dempster, A. P., N. M. Laird, D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistic Society* **Series B 39** 1–38.

Deneubourg, J.-L., S. Aron, S. Goss, J.-M. Pasteels. 1990. The self-organizing exploratory pattern of argentine ants. *Journal of Insect Behaviour* **3** 159–168.

Diks, E. B., A. G. de Kok, A. G. Lagodimos. 1996. Multi-echelon systems: A service-measure perspective. *European Journal of Operational Research* **95** 241–263.

Dittmar, D., J. Grahl, S. Klosterhalfen, S. Minner. 2007. Solving safety stock allocation problems using evolutionary algorithms. Technical Report 6/2007, University of Mannheim, Department of Logistics.

Domschke, W. 2007. *Logistik: Transport. Grundlagen, lineare Transport- und Umladeprobleme*. 5. Aufl. Oldenbourg, München.

Dorigo, M. 1992. Optimization, learning and natural algorithms. Ph.D. thesis, Dipartimento di Elletronica, Politecnico di Milano.

Dorigo, M., G. Di Caro. 1999. The ant colony optimization meta-heuristic. D. Corne, M. Dorigo, F. Glover, eds., *New Ideas in Optimization*. Advanced Topics in Computer Science, McGraw Hill, London, 11–32.

Dorigo, M., G. Di Caro, L. M. Gambardella. 1997. Ant coloncy system: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation* **1** 53–66.

Dorigo, M., T. Stützle, eds. 2004. *Ant Colony Optimization*. MIT Press, Cambridge, Massachusetts.

Drezner, Z., H. W. Hamacher, eds. 2002. *Facility Location, Applications and Theory*. Springer, Heidelberg, New York, Berlin.

Ducheyne, E. I., R. R. De Wulf, B. De Baets. 2002. Using linkage learning for forest management planning. Erick Cantú-Paz, ed., *Late Breaking papers at the Genetic and Evolutionary Computation Conference (GECCO-2002)*. AAAI, New York, 109–114.

Edmonds, J. 1976. Optimum branchings. *Journal of Research of the National Bureau of Standards* **71b** 233–240.

Eiben, A. E., J. E. Smith. 2007. *Introduction to Evolutionary Computing*. Natural Computing Series, Springer, Berlin, Heidelberg, New York.

Elmaghraby, S. E. 1960. Allocation under uncertainty when the demand has a continuous d. f. *Management Science* **6** 270–294.

Erenguc, S.S. 1988. Multiproduct dynamic lot-sizing model with coordinated replenishments. *Naval Research Logistics* **35** 1–22.

Erlenkotter, D. 1978. A dual-based procedure for uncapacitated facility location. *Operations Research* **26** 992–1009.

Etxeberria, R., P. Larrañaga. 1999. Global optimization using Bayesian networks. A. A. O. Rodriguez, M. R. S. Ortiz, R. S. Hermida, eds., *Proceedings of the Second Symposium on Artificial Intelligence CIMAF–1999*. Institute of Cybernetics, Mathematics and Physics, La Habana, Cuba, 332–339.

Federgruen, A., M. Tzur. 1991. A simple forward algorithm to solve general dynamic lot sizing models with n periods in O(n log n) or O(n) time. *Management Science* **37** 909–925.

Filipovic, V., J. Kratica, D. Tosic, I. Ljubic. 2000. Fine grained tournament selection for the simple plant location problem. *Proceedings of the 5th Online World Conference on Soft Computing in Industrial Applications WSC5*. Helsinki University of Technology, Helsinki, Finland, 152–158.

Fleischmann, B. 1993. Designing distribution systems with transport economies of scale. *European Journal of Operational Research* **70** 31–42.

Fleischmann, B. 2005. Distribution and transport planning. H. Stadtler, C. Kilger, eds., *Supply Chain Management and Advanced Planning. Concepts, Models, Software and Case Studies*. Springer, Berlin, Heidelberg, New York, 229–244.

Fleischmann, B., H. Meyr. 2003. Planning hierarchy, modeling and advanced planning systems. A. G. de Kok, S. C. Graves, eds., *Supply Chain Management: Design, Coordination and Operation*, *Handbooks in Operations Research and Management Science*, vol. 11. Elsevier, 457–523.

Friedman, N., M. Goldszmidt. 1996. Learning Bayesian networks with local structure. E. Horvits, F. Jensen, eds., *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence UAI–1996*. Morgan Kaufmann, San Francisco, California, 252–262.

Gallagher, M., M. Frean. 2005. Population-based continuous optimization, probabilistic modelling and the mean shift. *Evolutionary Computation* **13** 29–42.

Gallagher, M., M. Frean, T. Downs. 1999. Real–valued evolutionary optimization using a flexible probability density estimator. W. Banzhaf, et al., eds., *Proceedings of the Genetic and Evolutionary Computation Conference GECCO–1999*. Morgan Kaufmann, San Francisco, California, 840–846.

Galvão, R. D., L. A. Raggi. 1989. A method for solving to optimality uncapacitated location problems. *Annals of Operations Research* **18** 225–244.

Ghosh, D. 2003. Neighborhood search heuristics for the uncapacitated facility location problem. *European Journal of Operational Research* **150** 150–162.

Giacobini, M., A. Brabazon, S. Cagoni, G. A. Di Caro, R. Drechsler, M. Farooq, A. Fink, E. Lutton, P. Machado, S. Minner, M. O'Neill, F. Romero, J.and Rothlauf, G. Squillero, H. Takagi, A. S. Uyar, S. Yang, eds. 2007. *Applications of Evolutionary Computing*, *Lecture Notes in Computer Science*, vol. 4448. Springer, Berlin, Heidelberg, New York.

Glover, F., M. Laguna. 1997. *Tabu Search*. Kluwer Academic.

Goldberg, D. E. 1989. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, Massachusetts.

Goldberg, D. E. 2002. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, *Genetic Algorithms and Evolutionary Computation*, vol. 7. Kluwer Academic, Boston.

Goldberg, D. E., B. Korb, K. Deb. 1989. Messy genetic algorithms: Motivation, analysis and first results. *Complex Systems* **10** 385–408.

González, C., J. A. Lozano, P. Larrañaga. 2002. Mathematical modelling of UMDAc algorithm with tournament selection. Behaviour on linear and quadratic functions. *International Journal of Approximate Reasoning* **31** 313–340.

Grahl, J., P. A. N. Bosman, S. Minner. 2007a. Convergence phases, variance trajectories, and runtime analysis of continuous EDAs. D. Thierens, H.-G. Beyer, J. Bongard, J. Branke, J. A. Clark, D. Cliff, C. B. Congdon, K. Deb, B. Doerr, T. Kovacs, S. Kumar, J. F. Miller, J. Moore, F. Neumann, M. Pelikan, R. Poli, K. Sastry, K. O. Stanley, T. Stutzle, R. A. Watson, I. Wegener, eds., *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM Press, London, 516–522.

Grahl, J., P. A. N. Bosman, F. Rothlauf. 2006. The correlation-triggered adaptive variance scaling IDEA (CT-AVS-IDEA). M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. A. N. Bosman, M. V. Butz, C. Coello Coello, D. Dasgupta, S. G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan, D. Thierens, eds., *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM Press, Seattle, Washington, 397–404.

Grahl, J., S. Minner. 2006. Verteilungsschätzende Verfahren zur Lösung stochastischer Transportprobleme. M. Jacquemin, R. Pibernik, E. Sucky, eds., *Quantitative Methoden der Logistik und des Supply Chain Management*. Dr. Kovač, Hamburg, 339–353. In German.

Grahl, J., S. Minner, P. A. N. Bosman. 2007b. Learning structure illuminates black boxes - an introduction to estimation of distribution algorithms. Z. Michalewicz, P. Siarry, eds., *Advances in Metaheuristics for Hard Optimization*. Springer, Berlin, Heidelberg, New York, 309–340.

Grahl, J., S. Minner, F. Rothlauf. 2005. Behaviour of UMDAc with truncation selection on monotonous functions. D. Corne, Z. Michalewicz, B. McKay, G. Eiben, D. Fogel, C. Fonseca, G. Greenwood, G. Raidl, K. Chen Tan, A. Zalzala, eds., *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*. IEEE Press, Edinburgh, Scotland, 2553–2559.

Grahl, J., F. Rothlauf, S. Minner. 2008. Decomposition of single- and multiproduct lot-sizing problems and scalability of EDAs. A. Fink, F. Rothlauf, eds., *Advances in Computational Intelligence in Transportation and Logistics*. Series on Studies in Computational Intelligence, Springer, Berlin, Heidelberg, New York.

Grahl, J., T. Weiblen, S. Minner. 2007c. Linkage in warehouse location problems: The numbering effect and its influence on evolutionary algorithms. *Proceedings of the 2007 EURO Winter Institute on Location and Logistics*. 183–209.

Graves, S. C., S. P. Willems. 2000. Optimizing strategic safety stock placement in supply chains. *Manufacturing and Service Operations Management* **2** 68–83.

Graves, S. C., S. P. Willems. 2003. Supply chain design: Safety stock placement and supply chain configuration. A. G. de Kok, S. C. Graves, eds., *Supply Chain Management: Design, Coordination and Operation*. North-Holland, Amsterdam, 95–132.

Greene, W. H. 2003. *Econometric analysis*, vol. 5. Prentice Hall, Upper Saddle River.

Grünwald, P. 2005. Minimum description length tutorial. P. Grünwald, I. J. Myung, M. Pitt, eds., *Advances in Minimum Description Length: Theory and Applications*. MIT Press, Cambridge, Massachusetts, 23–81.

Guignard, M. 1988. A lagrangean dual ascent algorithm for simple plant location problems. *European Journal of Operational Research* **35** 193–200.

Hansen, N., S. D. Müller, P. Koumoutsakos. 2003. Reducing the time complexiy of the derandomized evolution strategy with covariance matrix adaption. *Evolutionary Computation* **11** 1–18.

Hansen, N., A. Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* **9** 159–195.

Harik, G. 1997. Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. Ph.D. thesis, University of Michigan.

Harik, G. 1999. Linkage learning via probabilistic modeling in the ECGA. Technical Report 99010, IlliGAL, University of Illinois, Urbana, Illinois.

Harik, G., E. Cantú-Paz, D. E. Goldberg, B. L. Miller. 1999. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation* **7** 231–253.

Harik, G., D. E. Goldberg. 1997. Learning linkage. R. K. Belew, M. D. Vose, eds., *Foundations of Genetic Algorithms 4*. Morgan Kaufmann, San Francisco, California, 247–262.

Harik, G., D. E. Goldberg. 2000. Linkage learning through probabilistic expression. *Computer Methods in Applied Mechanics and Engineering* **186** 295–310.

Harik, G., F. Lobo. 1999. A parameter-less genetic algorithm. W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, R. E. Smith, eds., *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*. Morgan Kaufmann, Orlando, Florida, USA, 258–265.

Harik, G., F. Lobo, D. E. Goldberg. 1998. The compact genetic algorithm. D. Fogel, ed., *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*. IEEE Press, Piscataway, New Jersey, 523–528.

Heckerman, D., D. Geiger. 1995. Learning Bayesian networks: A unification for discrete and Gaussian domains. P. Besnard, S. Hanks, eds., *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence UAI–1995*. Morgan Kaufmann, San Mateo, California, 274–284.

Heckerman, D., D. Geiger, D. M. Chickering. 1994. Learning Bayesian networks: The combination of knowledge and statistical data. R. L. de Mantaras, D. Poole, eds., *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence UAI–1994*. Morgan Kaufmann, San Mateo, California, 293–301.

Hillier, F. S., G. J. Lieberman. 2005. *Introduction to Operations Research*. 8th ed. McGraw-Hill, New York.

Hoefer, M. 2002. Performance of heuristic and approximation algorithms for the uncapacitated facility location problem. Tech. Rep. MPI-I-2002-1-005, Max-Planck-Institut für Informatik, Saarbrücken, Germany.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan.

Holmberg, K. 1995. Efficient decomposition and linearization methods for the stochastic transportation problem. *Computational Optimization and Applications* **4** 293–316.

Holmberg, K., K. O. Jörnsten. 1984. Cross decomposition applied to the stochastic transportation problem. *European Journal of Operational Research* **17** 361–368.

Holmberg, K., H. Tuy. 1999. A production-transportation problem with stochastic demand and concave production costs. *Mathematical Programming* **5** 157–179.

Horng, J. T., L. Y. Liu, B. Y. Lin, C. Y. Kao. 1999. Resolution of the uncapacitated warehouse location problem using a simple genetic algorithm. P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, A. Zalzala, eds., *Proceedings of the Congress on Evolutionary Computation*. IEEE Press, Washington D.C., 1186–1193.

Horst, R., H. Tuy. 1998. *Global Optimization*. 3rd ed. Springer, Berlin, Heidelberg, New York.

Humair, S., S. P. Willems. 2006. Optimizing strategic safety stock placement in supply chains with clusters of commonality. *Operations Research* **54** 725–742.

Jans, R., Z. Degraeve. 2007. Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research* **177** 1855–1875.

Jaramillo, J.H., J. Bhadury, R. Batta. 2002. On the use of genetic algorithms to solve location problems. *Computers & Operations Research* **29** 761–779.

Johnson, N. L., S. Kotz, N Balakrishnan, eds. 1994. *Continuous Univariate Distributions*, *Wiley Series in Probability and Mathematical Statistics*, vol. 1. 2nd ed. Wiley-Interscience, New York.

Joneja, D. 1990. The joint replenishment problem: New heuristics and worst case performance bounds. *Operations Research* **38** 711–723.

Jordan, M. I. 1999. *Learning in Graphical Models*. MIT Press, Cambridge, Massachusetts.

Judd, K. 1998. *Numerical Methods in Economics*. MIT Press, Cambridge, Massachusetts.

Jägersküpper, J. 2005. Rigorous runtime analysis of the (1+1)-ES: 1/5-rule and ellipsoidal fitness landscapes. A. H. Wright, M. D. Vose, K. A. De Jong, L. M. Schmitt, eds., *Foundations of Genetic Algorithms 8*. Springer, Berlin, Heidelberg, New York, 260–281.

Jägersküpper, J., C. Witt. 2005. Rigorous runtime analysis of a (μ+1)-ES for the sphere function. H.-G. Beyer, U.-M. O'Reilly, D. V. Arnold, W. Banzhaf, C. Blum, E. W. Bonabeau, E. Cantú-Paz, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. R. Raidl, T. Soule, A. M. Tyrrell, J.-P. Watson, E. Zitzler, eds., *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*. ACM Press, Washington DC, 849–856.

Kallel, L., B. Naudts, C. R. Reeves. 2001. Properties of fitness functions and search landscapes. B. Naudts L. Kallel, A. Rogers, eds., *Theoretical Aspects of Evolutionary Computing*. Springer, Berlin, 175–206.

Kao, E. P. C. 1979. A multi-product dynamic lot-size model with individual and joint setup costs. *Operations Research* **27** 279–289.

Kenyon, A. S., D. P. Morton. 2002. A survey on stochastic location and routing problems. *Central European Journal of Operations Research* **9** 277–328.

Kern, S., S. D. Müller, N. Hansen, D. Büche, J. Ocenasek, P. Koumoutsakos. 2004. Learning probability distributions in continuous evolutionary algorithms — a comparative review. *Natural Computing* **3** 77–112.

Khan, N., D. E. Goldberg, M. Pelikan. 2002. Multi–objective Bayesian optimization algorithm. Technical Report 2002009, IlliGAL, University of Illinois, Urbana, Illinois.

Khouja, M. 1999. The single-period (news-vendor) problem: literature review and suggestions for future research. *Omega* **27** 537–553.

Kimball, G. E. 1988. General principles of inventory control. *Journal of Manufacturing and Operations Management* **1** 119–130.

Kirkpatrick, S., C. D. Gelatt, M.P. Vecci. 1983. Optimization by simulated annealing. *Science* **220** 671–680.

Klose, A. 2001. *Standortplanung in distributiven Systemen: Modelle, Methoden, Anwendungen*. Physica-Verlag, Heidelberg.

Klose, A., A. Drexl. 2005. Facility location models for distribution system design. *European Journal of Operational Research* **162** 4–29.

Kochetov, Y., D. Ivanenko. 2003. Computationally difficult instances for the uncapacitated facility location problem. *Proceedings of the Fifth Metaheuristics International Conference (MIC2003)*. Kyoto, Japan, 351–364.

Kotz, S., N. Balakrishnan, N. L. Johnson. 2000. *Continuous Multivariate Distributions*, *Wiley Series in Probability and Mathematical Statistics*, vol. 2. Wiley-Interscience, New York.

Krarup, J., P. M. Pruzan. 1983. The simple plant location problem: Survey and synthesis. *European Journal of Operational Research* **12** 36–81.

Kratica, J., V. Filipovic, V. Sesum, D. Tosic. 1996. Solving the uncapacitated warehouse location problem by SGA with add-heuristic. *Proceedings of the XIV International Conference on Material handling and warehousing*. Belgrade, Serbia, 3.33–3.37.

Kratica, J., D. Tosic, V. Filipovic, I. Ljubic. 2001. Solving the simple plant location problem by genetic algorithm. *RAIRO Operations Research* **35** 127–142.

Kuehn, A.A., M.J. Hamburger. 1963. A heuristic program for locating warehouses. *Management Science* **9** 643–666.

Kullback, S., R. A. Leibler. 1951. On information and sufficiency. *Annals of Mathematical Statistics* **22** 79–86.

Larrañaga, P., R. Etxeberria, J. A. Lozano, J. M. Peña. 2000a. Optimization in continuous domains by learning and simulation of Gaussian networks. A. S. Wu, ed., *Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*. 201–204.

Larrañaga, P., R. Etxeberria, J. A. Lozano, J. M. Peña. 2000b. Optimization in continuous domains by learning and simulation of Gaussian networks. A. S. Wu, ed., *Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*. Las Vegas, Nevada, 201–204.

Larrañaga, P., J. Lozano, eds. 2001. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, *Genetic Algorithms and Evolutionary Computation*, vol. 2. Kluwer Academic Publishers, London.

Larrañaga, P., J. A. Lozano, V. Robles, A. Mendiburu, P. de Miguel. 2002. Searching for the best permutation with estimation of distribution algorithms. H. H. Hoos, T. Stuetzle, eds., *Proceedings of the Workshop on Stochastic Search Algorithms at the IJCAI–2001*. Morgan Kaufmann, San Francisco, California, 7–14.

Laumanns, M., J. Ocenasek. 2002. Bayesian optimization algorithms for multi–objective optimization. J. J. Merelo Guervós, P. Adamidis, H.-G. Beyer, J. L. F. Martín, H.-P. Schwefel, eds., *Parallel Problem Solving from Nature - PPSN VII, 7th International Conference*, *Lecture Notes in Computer Science*, vol. 2439. Springer, 298–307.

Lauritzen, S. L. 1996. *Graphical Models*. Clarendon Press, Oxford.

LeBlanc, L. J., R. V. Helgason, D. E. Boyce. 1985. Improved efficiency of the frank-wolfe algorithm for convex network programs. *Transportation Science* **19** 445–462.

Lesnaia, E. 2004. Optimizing safety stock placement in general supply chains. Ph.D. thesis, Massachusetts Institute of Technology.

Lima, C. F., F. G. Lobo. 2004. Parameter-less optimization with the extended compact genetic algorithm and iterated local search. K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P. L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, A. Tyrrell, eds., *Genetic and Evolutionary Computation – GECCO-2004*, *Lecture Notes in Computer Science*, vol. 3102. Springer, Seattle, Washington, 1328–1339.

Lima, C. F., M. Pelikan, K. Sastry, M. Butz, D. E. Goldberg, F. G. Lobo. 2006. Substructural neighborhoods for local search in the bayesian optimization algorithm. T. P. Runarsson, H.-G. Beyer, E. K. Burke, J. J. M. Guervós, L. D.

Whitley, X. Yao, eds., *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference*. No. 4193 in Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, New York, 232–241.

Lobo, F. G., C. F. Lima, Z. Michalewicz, eds. 2007. *Parameter Setting in Evolutionary Algorithms*, *Studies in Computational Intelligence*, vol. 54. Springer, Berlin, Heidelberg, New York.

Magnanti, T. L., Z.-J. M Shen, J. Shu, D. Simchi-Levi, C.-P. Teo. 2006. Inventory placement in acyclic supply chain networks. *Operations Research Letters* **34** 228–238.

Mandl, F. 1988. *Statistical Physics*. 2nd ed. The Manchester Physics Series, Wiley, New York.

Merz, P., B. Freisleben. 1999. Fitness landscapes and memetic algorithm design. D. Corne, M. Dorigo, F. Glover, eds., *New Ideas in Optimization*. McGraw Hill, 245–260.

Michalewicz, Z., D. B. Fogel. 2004. *How to Solve it: Modern Heuristics*, vol. 2nd. Springer.

Michel, L., P. van Hentenryck. 2004. A simple tabu search for warehouse location. *European Journal of Operational Research* **157** 576–591.

Minner, S. 1997. Dynamic programming algorithms for multi-stage safety stock optimization. *OR Spektrum* **19** 261–271.

Minner, S. 2000. *Strategic Safety Stocks in Supply Chains*. Springer, Berlin, Heidelberg, New York.

Minner, S., D. Dittmar, S. Klosterhalfen. 2006. A dynamic programming algorithm for safety stock optimization in general supply networks. Working Paper 07/2006, University of Mannheim.

Mühlenbein, H., T. Mahnig. 1999. FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation* **7** 353–376.

Mühlenbein, H., R. Höns. 2005. The estimation of distributions and the mimimum relative entropy principle. *Evolutionary Computation* **13** 1–27.

Mühlenbein, H., T. Mahnig, A. O. Rodriguez. 1999. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics* **5** 215–247.

Mühlenbein, H., G. Paaß. 1996. From recombination of genes to the estimation of distributions I. Binary parameters. H.-M. Voigt, W. Ebeling, I. Rechenberg, H.-P. Schwefel, eds., *Parallel Problem Solving from Nature - PPSN IV, International Conference on Evolutionary Computation.*, *Lecture Notes in Computer Science*, vol. 1141. Springer, 178–187.

Mühlenbein, H., D. Schlierkamp-Voosen. 1993. Predictive models for the breeder genetic algorithm. *Evolutionary Computation* **1** 25–49.

Narayanan, A., E. P. Robinson. 2006. More on 'models and formulations for the dynamic joint replenishment problem'. *International Journal of Production Research* **44** 383–297.

Ocenasek, J., S. Kern, N. Hansen, P. Koumoutsakos. 2004. A mixed Bayesian optimization algorithm with variance adaptation. X. Yao, E. Burke, J. A. Lozano, J. Smith, J. J. Merelo Guervos, J. A. Bullinaria, J. Rowe, P. Tino, A. Kaban, H. P. Schwefel, eds., *Parallel Problem Solving from Nature – PPSN VIII*. Springer, Berlin, 352–361.

Ocenasek, J., J. Schwarz. 2002. Estimation of distribution algorithm for mixed continuous-discrete optimization problems. *2nd Euro-International Symposium on Computational Intelligence*. 227–232.

Pardalos, P. M., J. B. Rosen. 1984. *Constrained Global Optimization: Algorithms and Applications*. Springer, Berlin, Heidelberg, New York.

Paul, T., H. Iba. 2003a. Real–coded estimation of distribution algorithm. *Proceedings of the 5th Metaheuristics International Conference 2003 — MIC–2003*. 1–6.

Paul, T. P., H. Iba. 2003b. Reinforcement learning estimation of distribution algorithm. E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, J. Miller, eds., *Genetic and Evolutionary Computation – GECCO-2003*, *Lecture Notes in Computer Science*, vol. 2723. Springer, Chicago, 1259–1270.

Pelikan, M. 2002. Bayesian optimization algorithm: From single level to hierarchy. Ph.D. thesis, University of Illinois at Urbana-Champaign, Dept. of Computer Science, Urbana, IL.

Pelikan, M., D. E. Goldberg. 2001. Escaping hierarchical traps with competent genetic algorithms. L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, E. Burke, eds., *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. Morgan Kaufmann, San Francisco, California, 511–518.

Pelikan, M., D. E. Goldberg. 2003. Hierarchical BOA solves ising spin glasses and MAXSAT. E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, J. Miller, eds., *Genetic and Evolutionary Computation – GECCO-2003*, *Lecture Notes in Computer Science*, vol. 2723. Springer, Chicago, 1271–1282.

Pelikan, M., D. E. Goldberg, E. Cantú-Paz. 1999. BOA: The Bayesian optimization algorithm. W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, R. E. Smith, eds., *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*. Morgan Kaufmann, Orlando, Florida, USA, 525–532.

Pelikan, M., D. E. Goldberg, E. Cantú-Paz. 2000. Bayesian optimization algorithm, population sizing, and time to convergence. D. Whitley, D. E. Goldberg, E. Cantù-Paz, L. Spector, I. Parmee, H.-G. Beyer, eds., *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. Morgan Kaufmann, Las Vegas, Nevada, 275–282.

Pelikan, M., D. E. Goldberg, K. Sastry. 2001. Bayesian optimization algorithm, decision graphs and occam's razor. L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, E. Burke, eds., *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. Morgan Kaufmann, San Francisco, California, 519–526.

Pelikan, M., H. Mühlenbein. 1998. Marginal distribution in evolutionary algorithms. *Proceedings of the International Conference on Genetic Algorithms Mendel '98*. Brno, Czech Republic, 90–95.

Pelikan, M., H. Mühlenbein. 1999. The bivariate marginal distribution algorithm. R. Roy, T. Furuhashi, P. K. Chawdhry, eds., *Advances in Soft Computing - Engineering Design and Manufacturing*. Springer, London, 521–535.

Pelikan, M., K. Sastry, M. V. Butz, D. E. Goldberg. 2006a. Hierarchical BOA on random decomposable problems. IlliGAL Report 2006002, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, IL.

Pelikan, M., K. Sastry, E. Cantú-Paz, eds. 2006b. *Scalable optimization via probabilistic modeling: From algorithms to applications*. Natural Computing Series, Springer, Berlin, Heidelberg, New York.

Pelikan, M., K. Sastry, D. E. Goldberg. 2003. Scalability of the bayesian optimization algorithm. *International Journal of Approximate Reasoning* **31** 221–258.

Pelikan, M., K. Sastry, D. E. Goldberg. 2005. Multiobjective hBOA, clustering, and scalability. H.-G. Beyer, U.-M. O'Reilly, D. V. Arnold, W. Banzhaf, C. Blum, E. W. Bonabeau, E. Cantú-Paz, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. R. Raidl, T. Soule, A. M. Tyrrell, J.-P. Watson, E. Zitzler, eds., *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*. ACM Press, Washington DC, 663–670.

Pelikan, M., Lin. T. 2004. Parameter-less hierarchical BOA. K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P. L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, A. Tyrrell, eds., *Genetic and Evolutionary Computation – GECCO-2004*, *Lecture Notes in Computer Science*, vol. 3103. Springer, Seattle, Washington, 24–35.

Pelikan, Martin, Kumara Sastry, David E. Goldberg. 2006c. Sporadic model building for efficiency enhancement of hierarchical BOA. M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. A. N. Bosman, M. V. Butz, C. Coello Coello, D. Dasgupta, S. G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan, D. Thierens, eds., *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM Press, Seattle, Washington, 405–412.

Qi, L. 1985. Forest iteration method for stochastic transportation problem. *Mathematical Programing Study* **25** 142–163.

Quinlan, J. 1993. Combining instance-based and model-based learning. *Proceedings of the Tenth International Conference on Machine Learning (ML-93)*. Morgan Kaufmann, San Mateo, California, 236–243.

Rastegar, R., M. R. Meybodi. 2005. A study on the global convergence time complexity of estimation of distribution algorithms. G. Wang, M. Szczuka, I. Duentsch, Y. Yao, D. Slezak, eds., *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing: 10th International Conference RSFDGrC-2005*, *Lecture Notes in Computer Science*, vol. 3641. Springer, Berlin, Heidelberg, New York, 441–450.

Rechenberg, I. 1973. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann–Holzboog, Stuttgart.

Robinson, E. P., L.-L. Gao. 1996. A dual ascent procedure for multiproduct dynamic demand coordinated replenishment with backlogging. *Management Science* **42** 1556–1564.

Robinson, E. P., A. Narayanan, L.-L. Gao. 2007. Effective heuristics for the dynamic joint replenishment problem. *Journal of the Operational Research Society* **58** 808–815.

Robles, V., P. de Miguel, P. Larrañaga. 2001. Solving the traveling salesman problem with EDAs. P. Larrañaga, J. A. Lozano, eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation.*, *Genetic and Evolutionary Computation*, vol. 2. Kluwer Academic Publishers, London, 211–230.

Rothlauf, F. 2006. *Representations for Genetic and Evolutionary Algorithms*. 2nd ed. Studies in Fuzziness and Soft Computing, Springer, Heidelberg, New York.

Rudlof, S., M. Köppen. 1996. Stochastic hill climbing with learning by vectors of normal distributions. T. Furuhashi, ed., *Proceedings of the First Online Workshop on Soft Computing (WSC1)*. Nagoya University, Nagoya, Japan, 60–70.

Salustowicz, R. P., J. Schmidhuber. 1997. Probabilistic incremental program evolution. *Evolutionary Computation* **5** 123–141.

Sastry, K., L. de la Ossa, F. G. Lobo. 2006. X-ary Extended Compact Genetic Algorithm in C++. IlliGAL report 2006013, University of Illinois at Urbana Champaign.

Sastry, K., D. E. Goldberg. 2003. Probabilistic model building and competent genetic programming. R. Riolo, B. Worzel, eds., *Genetic Programming Theory and Practice*. Kluwer Academic, Boston, Massachusetts, 205–220.

Sastry, K., M. Pelikan, D. E. Goldberg. 2004. Efficiency enhancement of probabilistic model building algorithms. R. Poli, S. Cagnoni, M. Keijzer, E. Costa, F. Pereira, G. Raidl, S. C. Upton, D. Goldberg, H. Lipson, E. de Jong, J. Koza, H. Suzuki, H. Sawai, I. Parmee, M. Pelikan, K. Sastry, D. Thierens, W. Stolzmann, P. L. Lanzi, S. W. Wilson, M. O'Neill, C. Ryan, T. Yu, J. F. Miller, I. Garibay, G. Holifield, A. S. Wu, T. Riopka, M. M. Meysenburg, A. W. Wright, N. Richter, J. H. Moore, M. D. Ritchie, L. Davis, R. Roy, M. Jakiela, eds., *GECCO 2004 Workshop Proceedings*. Seattle, Washington.

Schilling, D. A., K. E. Rosing, C. S. ReVelle. 2000. Network distance characteristics that affect computational effort in p-median location problems. *European Journal of Operational Research* **127** 525–536.

Sebag, M., A. Ducoulombier. 1998. Extending population–based incremental learning to continuous search spaces. A. E. Eiben, T. Bäck, M. Schoenauer,

H. P. Schwefel, eds., *Parallel Problem Solving from Nature – PPSN V*. Springer, Berlin, 418–427.

Servet, I., L. Trave-Massuyes, D. Stern. 1997. Telephone network traffic overloading diagnosis and evolutionary computation technique. J. K. Hao, E. Lutton, E. M. A. Ronald, M. Schoenauer, D. Snyers, eds., *Proceedings of Artificial Evolution '97*. Springer, Berlin, 137–144.

Shin, S.-Y., D.-Y. Cho, B.-T. Zhang. 2001. Function optimization with latent variable models. A. Ochoa, ed., *Proceedings of the Third International Symposium on Adaptive Systems ISAS–2001 — Evolutionary Computation and Probabilistic Graphical Models*. Institute of Cybernetics, Mathematics and Physics, 145–152.

Shin, S.-Y., B.-T. Zhang. 2001. Bayesian evolutionary algorithms for continuous function optimization. *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*. IEEE Press, Seoul, Korea, 508–515.

Sierra, B., E. Lazkano, I. Inza, M. Merino, P. Larrañaga, J. Quiroga. 2001. Prototype selection and feature subset selection by estimation of distribution algorithms. a case study in the survival of cirrhotic patients treated with tips. A. L. Rector, S. Quaglini, P. Barakona, S. Andreassen, eds., *Proceedings of the 8th Artificial Intelligence in Medicine in Europe AIME–2001*. Springer, Berlin, 20–29.

Silver, E. A., D. F. Pyke, R. Peterson. 1998. *Inventory Management and Production Planning and Scheduling*. 3rd ed. Wiley, New York, New York, USA.

Silver, E.A. 1979. Coordinated replenishments of items under time varying demand: Dynamic programming formulation. *Naval Research Logistics Quarterly* **26** 141–151.

Simpson, K. F. 1958. In-process inventories. *Operations Research* **6** 863–873.

Syswerda, G. 1993. Simulated crossover in genetic algorithms. L. D. Whitley, ed., *Proceedings of the Second Workshop on Foundations of Genetic Algorithms*. Morgan Kaufmann, San Mateo, California, 239–255.

Tatsuoka, M. M. 1971. *Multivariate Analysis: Techniques for Educational and Psychological Research*. John Wiley & Sons Inc., New York.

Thierens, D. 1995. Analysis and design of genetic algorithms. Ph.D. thesis, Leuven, Belgium: Katholieke Universiteit Leuven.

Thierens, D. 1999. Scalability problems of simple genetic algorithms. *Evolutionary Computation* **7** 331–352.

Thierens, D., D. E. Goldberg. 1993. Mixing in genetic algorithms. S. Forrest, ed., *Proceedings of the fifth conference on Genetic Algorithms*. Morgan Kaufmann, 38–45.

Tsutsui, S. 2002. Probabilistic model–building genetic algorithms in permutation representation domain using edge histogram. J. J. Merelo Guervós, P. Adamidis, H.-G. Beyer, J. L. F. Martín, H.-P. Schwefel, eds., *Parallel Problem Solving from Nature - PPSN VII, 7th International Conference*, *Lecture Notes in Computer Science*, vol. 2439. Springer, 224–233.

van Houtum, G. J., K. Inderfurth, W. H. M. Zijm. 1996. Material coordination in stochastic multi-echelon systems. *European Journal of Operational Research* **95** 1–23.

Veinott, A.F. 1969. Minumum concave cost solutions of Leontief substitution models of multi-facility inventory systems. *Operations Research* **17** 262–291.

Wagelmans, A., S. van Hoesel. 1992. Economic lot sizing: An O(n log n) algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research* **40** 145–156.

Wagner, H. M., T. M. Whitin. 1958. Dynamic version of the economic lot size model. *Management Science* **5** 89–96.

Wemmerlöv, U. 1982. A comparison of discrete single stage lot-sizing heuristics with special emphasis on rules based on the marginal cost principle. *Engineering Costs and Production Economics* **7** 45–53.

Williams, A. C. 1963. A stochastic transportation problem. *Operations Research* **11** 759–770.

Wilson, D. 1975. A mean cost approximation for transportation problems with stochastic demand. *Naval Research Logistics Quarterly* **22** 181–187.

Witten, P., H.-G. Zimmermann. 1978. Stochastische Transportprobleme. *Zeitschrift für Operations Research* **22** 55–68.

Yuan, B., M. Gallagher. 2005. On the importance of diversity maintenance in estimation of distribution algorithms. H.-G. Beyer, U.-M. O'Reilly, D. V. Arnold, W. Banzhaf, C. Blum, E. W. Bonabeau, E. Cantú-Paz, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. R. Raidl, T. Soule, A. M. Tyrrell, J.-P. Watson, E. Zitzler, eds., *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*. ACM Press, Washington DC, 719–726.

Yuan, B., M. Gallagher. 2006. A mathematical modelling technique for the analysis of the dynamics of a simple continuous EDA. M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. A. N. Bosman, M. V. Butz, C. Coello Coello, D. Dasgupta, S. G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan, D. Thierens, eds., *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM Press, Seattle, Washington, 1585–1591.

Zlochin, M., M. Birattari, N. Meuleau, M. Dorigo. 2004. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research* **131** 373–395.

Ich versichere, dass ich diese Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mannheim, den 12.12.2007

Jörn Grahl

Jörn Grahl

Ausbildung

Promotion:	Dr. rer. pol., „summa cum laude“ Universität Mannheim
Diplom:	Diplom-Kaufmann Universität Mannheim
Abitur:	Helene-Lange-Gymnasium Dortmund

Berufserfahrung

seit 01/2008	Post–Doc, Lehrstuhl für Wirtschaftsinformatik und BWL, Johannes Gutenberg-Universität Mainz
02/2005 – 08/2007	Wissenschaftlicher Mitarbeiter und Doktorand, Lehrstuhl für ABWL und Logistik, Universität Mannheim
2004	Promotionsstipendiat nach dem Landesgraduiertenförderungsgesetz, Lehrstuhl für Wirtschaftsinformatik I, Universität Mannheim
05/2002 – 07/2004	Wissenschaftliche Hilfskraft, Lehrstuhl für Wirtschaftsinformatik I, Universität Mannheim
2000 – 2003	Startup-Gründung und Geschäftsführung, NetTravelSystems GdbR Internetprogrammierung
04/2000 – 08/2000	Wissenschaftliche Hilfskraft, Lehrstuhl für Betriebswirtschaftslehre mit Schwerpunkt Fertigungswirtschaft

Qualifikationen

Kernkompetenzen:	Kombinatorische und kontinuierliche Optimierung in Produktion und Logistik, Simulation
Sprachkenntnisse:	Deutsch: Muttersprache Englisch: Verhandlungssicher Japanisch und Französisch: Grundkenntnisse
Softwarekenntnisse:	C++, Java, Perl, MATLAB, XpressMP, ARENA, SAP APO MS-Office, Gnuplot, Latex