On the Usefulness of Linkage Processing for Solving MAX-SAT

Krzysztof L. Sadowski
Institute of Information and Computing Sciences
Utrecht University
The Netherlands
K.L.Sadowski@uu.nl

Peter A. N. Bosman
Centre for Mathematics and Computer Science
Amsterdam
The Netherlands
Peter.Bosman@cwi.nl

Dirk Thierens
Institute of Information and Computing Sciences
Utrecht University
The Netherlands
D.Thierens@uu.nl

ABSTRACT

Mixing of partial solutions is a key mechanism used for creating new solutions in many Genetic Algorithms (GAs). However, this mixing can be disruptive and generate improved solutions inefficiently. Exploring a problem's structure can help in establishing less disruptive operators, leading to more efficient mixing. One way of using a problem's structure is to consider variable linkage information. Once a proper linkage model for a problem is obtained, mixing becomes more efficient.

This paper focuses on exploring different methods of building family of subsets (FOS) linkage models, which are then used with the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) to solve MAX-SAT problems. Individual algorithms from the GOMEA family are distinguished by how the FOS linkage models are constructed. The Linkage Tree Genetic Algorithm (LTGA) is a GOMEA instance which learns the linkage between problem variables by building a linkage tree in every generation. In this paper, we introduce SAT-GOMEA. This algorithm uses a predetermined FOS linkage model based on the SAT-problem's definition. Both algorithms use linkage information. We show that because of this information they are capable of performing significantly better than other algorithms from the GOMEA family which do not explore linkage. In a black-box (BBO) setting, LTGA performs well. We further study the use of linkage models outside of the typical BBO approach by examining the behavior of LTGA and the problem-specific SAT-GOMEA in a white-box setting, where more of the problem information is known. We show that with this white-box optimization (WBO) approach, exploring linkage information can still be beneficial.

We further compare the performance of these algorithms with a selection of non-GOMEA based algorithms. From the BBO perspective, we compare LTGA with the well-known hBOA. In the WBO setting, many very efficient problem-specific local search (LS) algorithm exist. We specifically consider Walksat and GSAT and show that combining LS with LTGA or SAT-GOMEA increases their performance.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Problem Solving, Search

General Terms

Algorithms, Performance, Experimentation

Keywords

Evolutionary Computation, Genetic Algorithms, Linkage Learning, Optimal Mixing Algorithm, Satisfiability

1. INTRODUCTION

How to best exploit a problem's structure is an ongoing research topic both in a black-box setting where no knowledge on the problem at hand is available and in a white-box setting where it is available. Linkage models, which attempt to describe meaningful dependencies between variables, can be powerful tools for the creation of new, highly-fit solutions. Mixing may be disruptive with respect to important building blocks otherwise [10]. A group of linkage models known as the family of subsets (FOS) has previously been introduced, and used with the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) framework [11]. The GOMEA framework provides an efficient mechanism for mixing partial solutions and generating new candidate solutions, given a FOS linkage model.

The FOS model represents the linkage between problem variables. Learning linkage and building a FOS model can be done in various ways including learning of variable dependencies or using a predetermined structure, when it can be derived. Within the GOMEA framework, we are interested in studying whether differently acquired FOS linkage models can have a significant impact on the fitness of generated solutions. In other words, when is the overhead of exploring linkage information useful ?

In this paper we study the Maximum Satisfiability (MAX-SAT) domain to analyze the effects of using different linkage models. The definition of the Satisfiability Problem (Section 2.2) permits us to take two separate approaches: black-box and white-box.

In the black-box setting, we assume no problem-specific information is known. We examine a GOMEA instance which dynamically learns the FOS linkage model by generating a linkage tree. This Linkage Tree Genetic Algorithm

(LTGA) has already been successfully used to solve problems such as the Nearest Neighbor NK-landscape or Deceptive Trap Functions with high efficiency [11]. We study whether the benefit of learning linkage extends to solving MAX-SAT problems. We also study the effects of complementing the LTGA with a local search, and contrast the results with another black-box algorithm, hBOA [4].

From the white-box optimization perspective, more of the SAT problem's information is exploitable. In addition to learning the linkage with LTGA, we introduce a new GOMEA instance: SAT-GOMEA. This algorithm predetermines the FOS linkage structure by exploiting the structural information from the SAT problem's formula definition. We show that for SAT problems known to posses an underlying structure, using linkage information yields significantly better solutions than for GOMEA instances which ignore linkage. We also present the benefits of combining LTGA and SAT-GOMEA with the well-known Walksat LS in the white-box setting.

While the main goal of this paper is to establish the significance of exploring variable linkage in the GOMEA instances, we also wish to study how our hybrid LTGA+Walksat and SAT-GOMEA+Walksat algorithms compare to SAT solvers Walksat and GSAT. Local search (such as GSAT, or Walksat) in the MAX-SAT domain have been reported by Rana and Whitley [7] to perform better than GAs. We wish to show how LTGA and SAT-GOMEA behave on both, uniformly distributed and structured MAX-SAT benchmarks, when combined with LS algorithms.

The remainder of this paper is organized as follows. In Section 2, we provide background information on linkage models and GOMEA. Then we define the Maximum Satisfiability problem and the benchmarks used in our experiments. In Section 3, we describe the black-box approach. Different FOS linkage model-building algorithms, as well as the experimental results are presented and discussed. In Section 4, we examine the white-box setting. We explain the SAT-GOMEA algorithm, and analyze the effects of different linkage models. We also provide experimental results of the hybrid LTGA+Walksat and SAT-GOMEA+Walksat algorithms. Section 5 summarizes and concludes our findings.

2. BACKGROUND

The Gene-pool Optimal Mixing Algorithm (GOMEA) is a framework which we use to explore the benefits of considering linkage information in the MAX-SAT domain. This section provides background on GOMEA and how the family of subsets (FOS) linkage model is used to create GOMEA instances. We also explain the MAX-SAT problem and present the SAT benchmarks used in our experiments.

2.1 Linkage Models and GOMEA

A linkage model has the capacity to represent groups of variables for a given problem, which make an important contribution to the fitness of solutions. In this paper, we use the family of subsets (FOS) linkage model to represent variable dependencies. A FOS is a set of subsets of a main set, which contains all problem variables [12]. Essentially each subset of the FOS contains between one and $l - 1$ items, where l is the number of problem variables. The number of subsets, their length and composition is determined based on the different FOS model-building algorithms. The difference between GOMEA instances used in this paper lie in how the FOS linkage model was determined.

GOMEA is a family of genetic algorithms which generate new solutions through mixing, given a FOS structure [12]. Initially, a random population of solutions is created and evaluated. Then, for each solution, the mixing process takes place. Each subset of the FOS model is used as the variation operator, or a mask, which is applied to copy values from a randomly picked donor solution from the population to the cloned solution and the effect is evaluated immediately. If for a given subset, the mask applied from the donor to the solution generates a higher-fitness solution, this new solution is kept. If it does not result in a better solution, the mask is rejected. For every solution, this mixing continues until all the FOS model masks have been tested. It guarantees that the generated candidate solutions are not worse than the original parent. If the solution is not improved after all the FOS masks are applied, Forced Improvement (FI) takes place [1]. The mixing process is repeated for this solution, but this time the donor is the best available solution in the population, instead of a random one. If this process also fails at improving the solution, this solution is entirely replaced with the best solution from the population. Ultimately, the set of newly generated solutions replaces the parent population entirely. Pseudo-code of this algorithm can be found in Figure 1.

```
GOMEA //population size n, problem size l
for $i \in \{0,1,\ldots,n-1\}$ do
    $\mathcal{P}_i \leftarrow$ CreateRandomSolution()
    EvaluateFitness($\mathcal{P}_i$)
while $\neg$TerminationCriterionSatisfied do
    $\boldsymbol{x}^{best} \leftarrow arg\ \max_{\boldsymbol{x}\in\mathcal{P}}\{fitness[\boldsymbol{x}]\}$
    $\mathcal{S} \leftarrow$ TournamentSelection($\mathcal{P}, n, 2$)
    LearnModel($\mathcal{S}$)
    for $i \in \{0,1,\ldots,n-1\}$ do
        $\mathcal{O}_i \leftarrow$ FI-GOM($\mathcal{P}_i$)
    $\mathcal{P} \leftarrow \mathcal{O}$
```

```
FI-GOM($\boldsymbol{x}$)
$\boldsymbol{b} \leftarrow \boldsymbol{o} \leftarrow \boldsymbol{x}$; $fitness[\boldsymbol{b}] \leftarrow fitness[\boldsymbol{o}] \leftarrow fitness[\boldsymbol{x}]$; $improved \leftarrow false$
for $i \in \{0,1,\ldots,|\mathcal{F}|-1\}$ do
    $\boldsymbol{p} \leftarrow$ Random($\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{n-1}\}$)
    $\boldsymbol{o}_{\boldsymbol{F}^i} \leftarrow \boldsymbol{p}_{\boldsymbol{F}^i}$
    if $\boldsymbol{o}_{\boldsymbol{F}^i} \neq \boldsymbol{b}_{\boldsymbol{F}^i}$ then
        EvaluateFitness($\boldsymbol{o}$)
        if $fitness[\boldsymbol{o}] > fitness[\boldsymbol{b}]$ then
            $\boldsymbol{b}_{\boldsymbol{F}^i} \leftarrow \boldsymbol{o}_{\boldsymbol{F}^i}$; $fitness[\boldsymbol{b}] \leftarrow fitness[\boldsymbol{o}]$; $improved \leftarrow true$
        else
            $\boldsymbol{o}_{\boldsymbol{F}^i} \leftarrow \boldsymbol{b}_{\boldsymbol{F}^i}$; $fitness[\boldsymbol{o}] \leftarrow fitness[\boldsymbol{b}]$
if $\neg improved$ then
  for $i \in \{0,1,\ldots,|\mathcal{F}|-1\}$ do
    $\boldsymbol{o}_{\boldsymbol{F}^i} \leftarrow \boldsymbol{x}^{best}_{\boldsymbol{F}^i}$
    if $\boldsymbol{o}_{\boldsymbol{F}^i} \neq \boldsymbol{b}_{\boldsymbol{F}^i}$ then
        EvaluateFitness($\boldsymbol{o}$)
        if $fitness[\boldsymbol{o}] > fitness[\boldsymbol{b}]$ then
            $\boldsymbol{b}_{\boldsymbol{F}^i} \leftarrow \boldsymbol{o}_{\boldsymbol{F}^i}$; $fitness[\boldsymbol{b}] \leftarrow fitness[\boldsymbol{o}]$; $improved \leftarrow true$
        else
            $\boldsymbol{o}_{\boldsymbol{F}^i} \leftarrow \boldsymbol{b}_{\boldsymbol{F}^i}$; $fitness[\boldsymbol{o}] \leftarrow fitness[\boldsymbol{b}]$
    if $improved$ then breakfor
if $\neg improved$ then
    $\boldsymbol{o} \leftarrow \boldsymbol{x}^{best}$; $fitness[\boldsymbol{o}] \leftarrow fitness[\boldsymbol{x}^{best}]$
```

Figure 1: Pseudo-code for GOMEA where FI-GOM is the genepool optimal mixing operator with forced improvement for a single solution.

2.2 Problem Definition

The Boolean Satisfiability (SAT) Problem is a well-known NP-complete problem. Its instance consists of a set of Boolean variables $X = \{x_1, x_2, x_3, .., x_l\}$, as well as a Boolean formula $F : \{0, 1\}^n \rightarrow \{0, 1\}$. This formula is a logical conjunction of clauses of the form $c_1 \wedge c_2 \wedge c_3 \wedge ... \wedge c_m$, where each clause c_i is a logical disjunction of a subset of the problem variables (or its negations).

An example clause would be $c_i = (x_1 \vee x_2 \vee \neg x_4)$ etc. An example of a complete formula with l=3 variables and m=3 clauses could look like $(\neg x_2) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2)$.

The objective in solving a SAT Problem is to find a variable assignment $v : X \rightarrow \{0, 1\}$, which will result in the Boolean formula evaluating to True. MAX-SAT is a generalization of the SAT problem. The objective of a MAX-SAT problem is to maximize the number of clauses which evaluate to True. In this paper the number of unsatisfied clauses (false clauses) is a measure of fitness, where zero refers to a fully satisfied formula. This means that in our experiments our goal is to minimize the fitness value.

2.2.1 SAT Benchmarks

To show whether linkage information can improve the fitness of generated solutions we need to examine benchmarks with both, little and lots of underlying structure. Benchmarks generated from a uniformly randomized distribution are expected to have very little exploitable structure in terms of variable dependencies. Other benchmarks, such as non-uniform random, handmade or industrial benchmarks are expected to have more structure.

Empirical data is gathered over a set of benchmarks available from the SAT Problem Library: SAT-Lib [3]. Differently sized benchmarks, in terms of number of variables and clauses are considered, as well as different benchmark types: random, handmade and industrial. Summary of all benchmarks can be found in Table 1. The 'unif' benchmark instance is the only instance generated randomly from a uniform distribution.

Benchmark	Abbr.	Vars.	Clauses	Type
uuf250-01	unif	250	1065	Random
sw100-X	sw100-X	500	3100	Random
am-5-5.shuffled-as.sat03-361	am-5-5	1076	3677	Industrial
hardnm-L32-03-S1527311839.shuffled-as.sat03-942	hard32	1024	4096	Random
C880mul.miter.shuffled-as.sat03-348	C880	1612	9373	Industrial
20000009987nc.shuffled-as.sat03-1665	2nc	2756	10886	Handmade

Table 1: Specifications of the MAX-SAT benchmarks used in this paper.

3. BLACK-BOX OPTIMIZATION

From the black-box optimization (BBO) perspective, problem-specific information is unknown. It is a generic perspective, where only the number of variables and the evaluation function are given. In this setting, the number of evaluations performed by any algorithm in order to reach better fitness values is a frequently used measure of performance. It should be noted however, that building a linkage model also requires computational effort. This effort may be non-negligible. However, it should also be noted that the LTGA requires a relatively small computational overhead of $O(nl^2)$, compared to most higher-order model-building evolutionary algorithms such as hBOA, which requires $O(nl^2+l^3)$ [5]. To study the behavior of different linkage models, we consider the linkage learning algorithm LTGA and contrast it against other GOMEA instances. We also compare our black-box results with the well-known hBOA EDA [4].

3.1 Algorithms

In this section, we examine how much influence learning linkage information has on the quality of generated solutions. To do this, we study the performance of GOMEA instances which determine the FOS linkage structure differently.

Before providing an overview of the BBO algorithms used, a small algorithmic change to the GOMEA's solution acceptance criteria is worth noting. Previous GOMEA instances often share the same solution acceptance criteria. A newly generated solution is only accepted and introduced to the population if this new candidate solution is strictly better than its parent. We replaced this condition with a less strict one. More specifically, a solution will replace its parent if it is at least as good. In terms of a SAT problem, this allows for more plateau exploration and diversification of the population, as equally good, but different solutions are not rejected. This change has been applied to all GOMEA instances discussed in this paper. Figure 2 exemplifies typical improvements achieved with this modification, which can be substantial.

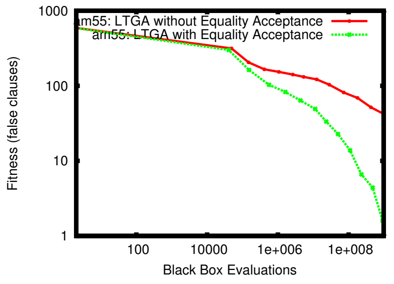

Figure 2: Typical improvement from accepting equally fit solutions during mixing.

3.1.1 LTGA

The Linkage Tree Genetic Algorithm is a GOMEA instance that learns the linkage between the problem variables, by generating a linkage tree at each generation. Building this tree is accomplished by a hierarchical clustering algorithm [10]. To measure dependencies between variables the hierarchical clustering algorithm uses mutual information. Based on this measure the linkage tree is build from the bottom up. Each variable is first assigned to a single cluster. Based on the mutual information new clusters are created by joining together already existing clusters. This process continues until only one cluster (containing all variables) is left, which requires an computational overhead of $O(nl^2)$ [2]. Each cluster of the linkage tree, except for the full cluster (which contains all variables), represents a subset of a FOS linkage model which is then used with GOMEA.

3.1.2 Random-Tree GOMEA

This GOMEA instance has a linkage structure identical to LTGA. While the FOS linkage structure is still based on a Linkage Tree, the actual linkage information is not based on learned variable dependencies. The hierarchical clustering algorithm used by LTGA is still present, and the linkage tree is still generated. However, the mutual information used to determine variable dependencies and cluster proximities is completely random. In other words, this GOMEA instance generates it's FOS structure similarly to LTGA in terms of mechanics, however no real linkage information is discovered.

3.1.3 Univariate GOMEA

Univariate GOMEA is a very simple instance of the GOMEA family. All the problem variables are considered to be independent of all other variables resulting in no linkage between the variables being modeled. In other words, the FOS linkage structure only consists of singleton subsets.

3.2 BBO Linkage Model Experiments

Figure 3 shows how different algorithms from the GOMEA family perform on our representative benchmarks. For every problem and every population size considered, we perform 30 independent runs. The only algorithm in the black-box setting which attempts to learn linkage information is LTGA.

The results show that learning linkage in a uniformly distributed random problem is not very efficient with small population sizes. Initially, Univariate-GOMEA finds better solutions with less evaluations. However, with large enough population, LTGA outperforms the other algorithms.

The results for problems with a better defined underlying structure favor linkage learning. In benchmarks which are not uniformly distributed the use of linkage information shows significant improvement, even for smaller population sizes. LTGA outperforms the other algorithms. This indicates that learning variable dependencies and including those dependencies in the linkage model can be a powerful tool in the MAX-SAT domain. These results only show the advantages in terms of evaluations needed to reach near-optimal solutions. We also consider the cost of the linkage learning overhead in terms of runtime.

LTGA is the most expensive GOMEA instance, because it requires the linkage learning overhead. Table 2 shows the fraction of the total runtime needed by learning to reach highly-fit solutions. The results show that the time needed for linkage learning is about ten percent of the total time needed by the algorithm. The other algorithms however, may never reach good solutions, or require at least multiple orders of magnitude more time to reach similar fitness levels on the tested benchmarks. This means that the overhead of learning is acceptable in terms of runtime limitations as well.

3.3 hBOA and LTGA Experiments

The hierarchical Bayesian optimization algorithm (hBOA) is a well-known EDA proposed by Pelikan and Goldberg [4] as a black-box approach to solving hierarchical and nearly decomposable problems. hBOA works with a population of solutions, which are initially random. Unlike GOMEA, each generation hBOA builds a Bayesian network as a probabilistic model of promising solutions. New solutions are generated though sampling the learned network [4]. While it is a competent algorithm in terms of linkage learning, the overhead of building a Bayesian network can be substantial.

Instance	BBO f	St.Dev f	WBO f	St.Dev f
unif	0.093	0.012	0.116	0.019
sw100-1	0.124	0.004	0.405	0.025
am5-5	0.097	0.007	0.342	0.020
hard32	0.061	0.011	0.301	0.033
c880	0.068	0.029	0.261	0.057
2nc	0.098	0.099	0.367	0.034

Table 2: Fraction f of total time needed by for linkage learning of the LTGA in BBO and WBO setting

hBOA was also tested on MAX-SAT. In their experiments, two local search algorithms are studied: GSAT and Walksat. GSAT is a best-improvement local search algorithm designed for the Satisfiability Problem [9]. Walksat is an extension of GSAT which incorporates random walks. Both of these algorithms are explained in more detail in Section 4. Pelikan and Goldberg show that hBOA combined with GSAT local search (hBOA+GSAT) outperforms the stand-alone GSAT and Walksat on the morphed graph-coloring problems, which are translated into MAX-SAT. These graph-coloring problems are known to be difficult for LS algorithms [4]. While GSAT and Walksat do not succeed at satisfying the tested problems within the tested runtime constrains, the hBOA+GSAT does.

In order to compare the effectiveness of LTGA, we examine a LTGA+GSAT hybrid on the same benchmark set as in [4]. It is important to note some differences between using GSAT with LTGA and with hBOA. With LTGA, a single GSAT instance is executed after mixing is complete for a given solution, which consists of many evaluations. Additionally, GSAT is allowed to run until no fitness-improving move is possible in l flips, where l is the number of problem variables. hBOA performs a GSAT search after every evaluation. It is unclear however, how many non-improving side-steps, if any, this GSAT is permitted to take.

Just like the hBOA+GSAT hybrid, LTGA+GSAT succeeded at solving the graph coloring benchmarks. Table 3 summarizes the results in terms of total evaluations needed. Each LTGA+GSAT result found in Table 3 represents the number of evaluations performed by LTGA, averaged over thirty independent runs on the smallest population size that is successful. The result show the number of evaluations of hBOA and LTGA respectively, not representing the flips performed by the local search.

4. WHITE-BOX OPTIMIZATION

Algorithms examined in the previous section attempted optimization without any problem-specific information. However, the Satisfiability domain provides us with more information. Examining the Boolean formula structure of a SAT Problem, as defined in Section 2, allows for many adjustments which can improve the quality and the speed of generating solutions. With careful bookkeeping, partial-solution evaluations and constant-time neighborhood search steps are possible, and can be used to improve the optimization algo-

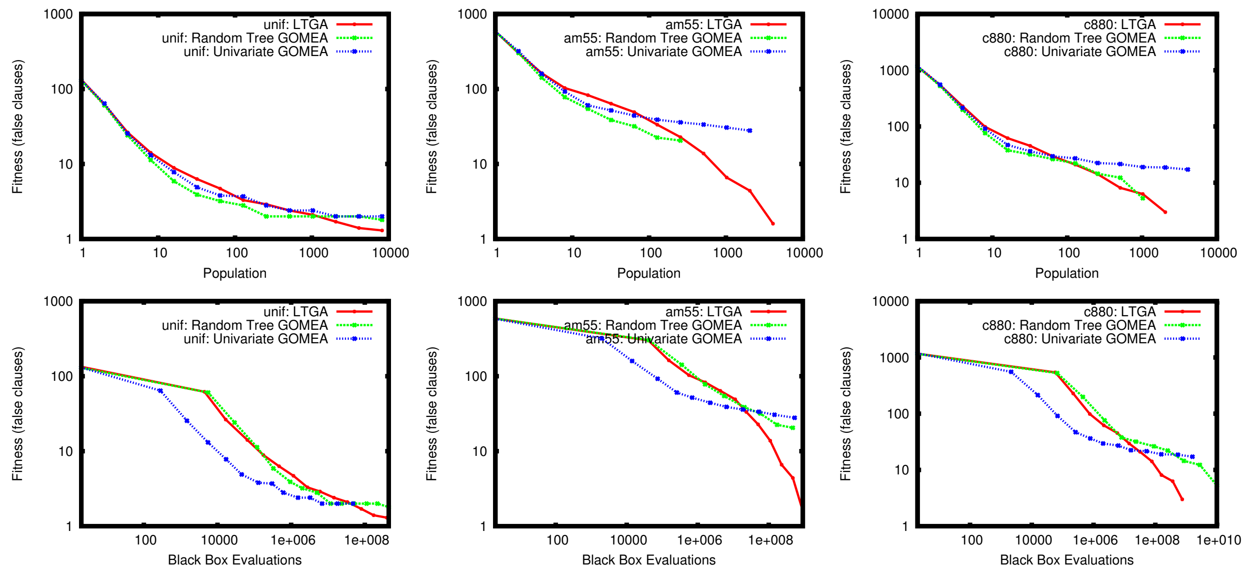

Figure 3: Comparison of performances of different GOMEA instances with different FOS Linkage Models on differently structured benchmarks.

Instance	hBOA+GSAT	LTGA+GSAT
SW100-8-5/sw100-1	1,262,018	257,823
SW100-8-5/sw100-2	1,099,761	209,972
SW100-8-5/sw100-3	1,123,012	232,916
SW100-8-6/sw100-1	1,183,518	622,748
SW100-8-6/sw100-2	1,324,857	1,388,747
SW100-8-6/sw100-3	1,629,295	1,948,879
SW100-8-7/sw100-1	1,732,697	1,433,359
SW100-8-7/sw100-2	1,558,891	1,354,856
SW100-8-7/sw100-6	1,966,648	2,659,001
SW100-8-7/sw100-7	1,222,615	917,577
SW100-8-8/sw100-1	1,219,675	1,110,319
SW100-8-8/sw100-2	1,537,094	766,796
SW100-8-8/sw100-6	1,650,568	968,124
SW100-8-8/sw100-7	1,287,180	1,322,188

Table 3: Number of hBOA and LTGA evaluations in a Black-Box setting

rithm [9]. When a variable is flipped, we only need to evaluate clauses which are affected by the flip, instead of performing a full evaluation.

Since full function evaluations are no longer needed in this setting, the performance measuring criteria is changed. We measure the performance of algorithms in the white-box setting by the number of bit flips performed. This means that a single move performed by a local search is equivalent to one flip, while a mixing operation on a mask of length l is considered to be l bit flips. Recall, in the black-box setting the number of bit flips in a mixing operator did not matter, as a full evaluation was needed every time.

In this section, we revisit the experiments performed in the black-box setting. Will learning of linkage remain beneficial in this setting ? We also introduce a new GOMEA instance, SAT-GOMEA, whose linkage structure is predetermined, and based on some problem-specific information.

The second aspect of our white-box experimentation will take advantage of relatively cheap local search algorithms, which can be used in the Satisfiability domain. Recall that in this setting making a best-improvement bit flip can be accomplished in constant time. Because of this, many successful local searchers have been developed for the SAT-Problem. We examine the GSAT and Walksat local search algorithms and compare them with LTGA+Walksat and SAT-GOMEA+Walksat hybrids.

4.1 Algorithms

In addition to the equal solution acceptance modification inherited from the black-box setting, another modification to all the GOMEA family algorithms tested in this section is introduced: path remembering. A crucial feature of all GOMEA instances is generating a Family of Subsets (FOS) linkage model. Those subsets specify a subdivision of all variables which is used in the mixing process. When applying a subset mask, every bit in the mask needs to be changed according to the material in the donor solution [12]. The path remembering modification allows for looking at every intermediate solution which was generated while applying a given subset mask. If one of the intermediate solutions is more fit than the one generated from applying the full mask, this intermediate solution is kept instead. Using path remembering is possible in the white-box setting, because the bit flip cost of evaluating a full mask is identical to traversing every bit flip of this mask individually in random order. This is not the case in the black-box setting. In the a black-box setting, using a full mask costs one full function evaluation, however performing each bit flip individually would require a full evaluation at every step. Figure 4 exemplifies the typical improvements achieved by introducing path remembering. Since the difference in terms of bit-flips used may not be clear on a log-scale graph, t-test is used to confirm statistical significance with p-value less then 0.001.

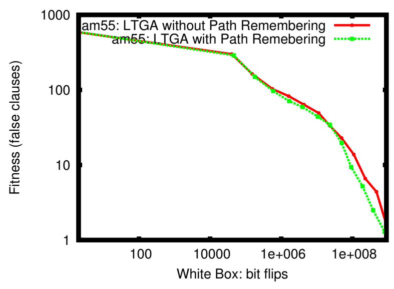

Figure 4: Typical improvement from allowing Path Remembering during solutions mixing in the white-box setting.

We study the linkage model effects from the white-box perspective using four different GOMEA instances. The first three have already been introduced in the black-box setting: LTGA, Random-Tree GOMEA and Univariate GOMEA. In this section we introduce another GOMEA instance: SAT-GOMEA, which generates a predetermined linkage model. We also explain two local search algorithms for solving SAT problems: GSAT and Walksat. Finally, we present a hybrid configuration of LTGA+Walksat and SAT-GOMEA+Walksat.

4.1.1 SAT-GOMEA

SAT-GOMEA introduces a predetermined linkage model. Is is a strictly white-box algorithm, as it requires problem-specific information which is not available in the black-box setting. SAT-GOMEA uses a predetermined FOS linkage model which is directly related to the SAT-Problem's Boolean formula. Specifically, the FOS linkage model of this algorithm consists of subsets mapped directly from the clauses of the Boolean formula. Each clause of the formula is represented by a subset in the linkage model and each variable in a clause is represented by the same variable within a subset. For example a clause $c_i = (x_1 \vee x_2 \vee \neg x_4)$ would be represented by a subset of the form $s_i = \{x_1, x_2, x_4\}$

4.1.2 Local Search Algorithms

GSAT and Walksat algorithms are well-known Satisfiability problem solvers. They are very simple and fast, yet surprisingly well-performing algorithms. GSAT is a best-improvement local search algorithm. If there is no actual improvement possible, a sideways step is taken by choosing a random bit flip which does not change the fitness value [9]. Walksat consists of a GSAT move and a walk move which are performed with probability p and $1 - p$ respectively. GSAT moves consist of performing a best possible (fitness-improving) single bit flip. While a GSAT move is not fully deterministic because tie breaks between different solutions with the same best fitness improvements and sideways steps are resolved randomly, it is the walk move that really brings a stochastic factor to the Walksat algorithm. During a walk move, a variable from a randomly chosen unsatisfied clause is flipped [8]. While the GSAT move attempts to always improve the current solution, the walk move allows for a greater search-space exploration. Throughout our experiments, we use the default p value of 0.5 for Walksat.

4.1.3 Hybrids

LTGA+Walksat and SAT-GOMEA+Walksat are two hybrid algorithms which combine modeling linkage information, genetic solution mixing and relatively inexpensive and efficient local search. After a solution is generated by the respective genetic algorithm, it is subject to a short local optimization performed by Walksat. We wish to show that on many benchmarks which contain underlying structures, using a hybrid algorithm can generate results competitive with stand-alone GSAT or Walksat in terms of best fitness found and the number of flips performed, within our experimentation limits.

4.2 WBO Linkage Model Experiments

We wish to examine if linkage information is still a factor from the white-box optimization perspective. Recall, in the black-box setting the Linkage Tree Genetic Algorithm performed significantly better than algorithms which did not use linkage information in their FOS structures. This time we have the LTGA and SAT-GOMEA algorithms which use some linkage information, and Random-Tree GOMEA and Univariate GOMEA which ignore the linkage. Results in Figure 5 correspond with the black-box results. On a uniformly generated random benchmark, exploring the linkage by learning, or using a predetermined structure is an unnecessary overhead with small population sizes. With large enough population budget however, benefits of linkage information become clear. On more challenging and structured benchmarks this advantage becomes clear sooner. Algorithms using linkage information find near-optimal solutions faster in terms of bit-flips.

We again also considered runtime. We measured how much the learning overhead adds to the runtime of the algorithms. LTGA is the most expensive GOMEA instance, because it requires the linkage learning overhead. Table 2 shows the fraction of the total runtime needed by learning to reach highly-fit solutions. The results show that learning becomes relatively more expensive in the white-box setting. This was anticipated, as the rest of the algorithm can perform faster in this setting. However, the time needed for learning remains a constant factor, which did not exceed more than 45 percent of the total runtime on any of the tested benchmarks. While it is possible to construct a benchmark where this conclusion would not hold (very many variables, very little clauses) , for most interesting benchmarks (near the phase-transition) the learning overhead should remain relatively small due to the clause-to-variable ratio. The results show that on those benchmarks GOMEA instances which do not use linkage information may never reach, or require multiple orders of magnitude more time in order to reach the same levels of fitness, again favoring the use of building linkage models, even in the white-box setting.

4.3 Hybrid Experiments

So far in the white-box setting we only described experiments performed with the GOMEA instances exclusively. While the results are promising, it does not help us understand how these algorithms perform in comparison to other SAT solvers. Figure 6 shows the LTGA and SAT-GOMEA algorithms enhanced with a Walksat LS. After any new solution was generated by one of those GAs, Walksat was allowed to optimize it further for $5 * l$ flips, where l is the number of variables in a problem. The $5 * l$ search length was chosen, as it generated good results and balanced the number of flips performed by the GA and local searcher well.

It is not a surprise that on many simple and uniformly structured problems the stand-alone Walksat performs better. In fact, even on more structured but relatively simple benchmarks is arrives at the optimal solution faster. However, as the stand-alone GSAT or Walksat algorithms fail to reach good solutions on more complex benchmarks, the LTGA and SAT-GOMEA hybrid versions continue to find better solutions with same number of bit-flips.

It remains very benchmark-specific which, LTGA or SAT-GOMEA, performs better in terms of required bit-flips. However, the results show that using linkage information in the GOMEA setting successfully helps to guide Walksat into promising search space locations. By doing so, Walkast can discover solutions which it failed to encounter on its own. Moreover, given that no additional computation overhead is required for building the linkage model in SAT-GOMEA, it is preferable or at least worth trying.

The Average Landscape-Guided Hopping (ALGH) algorithm was recently introduced for MAX-SAT [6]. ALGH uses a population of independent GSATs, whose results are averaged and generate a single solution. This solution is then used as starting point for Walksat. ALGH outperforms GSAT and Walksat, however reported experiments are limited to uniformly distributed MAX3SAT instances above the phase transition. Our experiments show that obtaining linkage information can be very useful on problems which are not uniform, and posses underlying structure.

5. CONCLUSIONS

We have discussed the behavior of instances from the Gene-pool Optimal Mixing Algorithm family in the MAX-SAT domain. The algorithms differ in the fashion in which the FOS Linkage Model was created. In the generic black-box optimization setting, LTGA, which obtains its linkage model by learning and constructing a linkage tree, was proven superior over other GOMEA instances which did not examine linkage on benchmarks of sufficient complexity.

We were unable to determine if the linkage structure predetermined with SAT-GOMEA performed better than one learned by LTGA. The results varied based on the benchmark tested. However, exploitation of linkage information proved to be just as beneficial in the white-box setting. Algorithms which build linkage models based on some structural knowledge, the linkage learning LTGA and the predetermined linkage SAT-GOMEA, performed better than GOMEA instances which lacked linkage information. This lets us conclude that for benchmarks with some underlying structure and sufficient difficulty, the exploration of the linkage generates better-fit solutions faster. The results hold in terms of evaluations performed in both black and white-box settings, and in terms of the total runtime the algorithms need to generate near-optimal solutions.

To compare those GAs with other existing solvers, we created a LTGA+Walksat and SAT-GOMEA+Walksat hybrids. LS algorithms (such as GSAT, and Walksat) in the MAX-SAT domain have been reported to perform better than GAs. Our results confirm that LTGA does not outperform algorithms such as Walksat on uniformly randomized MAX-SAT problems and other benchmarks which are easily solvable by local searchers. However, in cases of more structured and difficult benchmarks which cause local searchers to stall, both of our hybrid algorithms continued to improve their solutions.

Extensive research is ongoing in the SAT-problem domain. It is more than likely that for problems where the LTGA or SAT-GOMEA do not perform well, algorithms already exist which can solve these problems more efficiently. The optimal structure of a SAT problem for use within GOMEA is still not known, and research on further improvements is possible. We did however succeed in showing that exploring the structure of some satisfiability problems is beneficial. Furthermore, our results added evidence to the idea that the use of linkage information could be a useful optimization technique for a broader range of problems, both in the black-box and white-box setting.

6. REFERENCES

[1] P. A. N. Bosman and D. Thierens. Linkage neighbors, optimal mixing and forced improvements in genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '12, pages 585–592, New York, NY, USA, 2012. ACM.

[2] I. Gronau and S. Moran. Optimal implementations of UPGMA and other common clustering algorithms. *Information Processing Letters*, 104(9):205–210, 2007.

[3] H. H. Hoos and T. StÃijtzle. SATLIB: An Online Resource for Research on SAT. pages 283–292. IOS Press, 2000.

[4] M. Pelikan and D. E. Goldberg. Hierarchical BOA Solves Ising Spin Glasses and MAXSAT. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '03, pages 1271–1282. Springer, 2003.

[5] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz. BOA: the bayesian optimization algorithm. pages 525–532. Morgan Kaufmann, 1999.

[6] A. Prügel-Bennett and M. Tayarani-Najaran. Maximum satisfiability: Anatomy of the fitness landscape for a hard combinatorial optimization problem. *IEEE transactions on evolutionary computation*, 16(3):319–338, 2012.

[7] S. Rana and D. Whitley. Genetic algorithm behavior in the maxsat domain. In *Parallel Problem Solving from Nature*, pages 785–794. Springer, 1998.

[8] B. Selman, H. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 521–532, 1995.

[9] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of AAAI-92*, pages 440–446, 1992.

[10] D. Thierens. The linkage tree genetic algorithm. In *Parallel Problem Solving from Nature — PPSN XI*, pages 264–273, Berlin, 2010. Springer–Verlag.

[11] D. Thierens and P. A. N. Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '11, pages 617–624, New York, NY, USA, 2011. ACM.

[12] D. Thierens and P. A. N. Bosman. Predetermined versus learned linkage models. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '12, pages 289–296, New York, NY, USA, 2012. ACM.

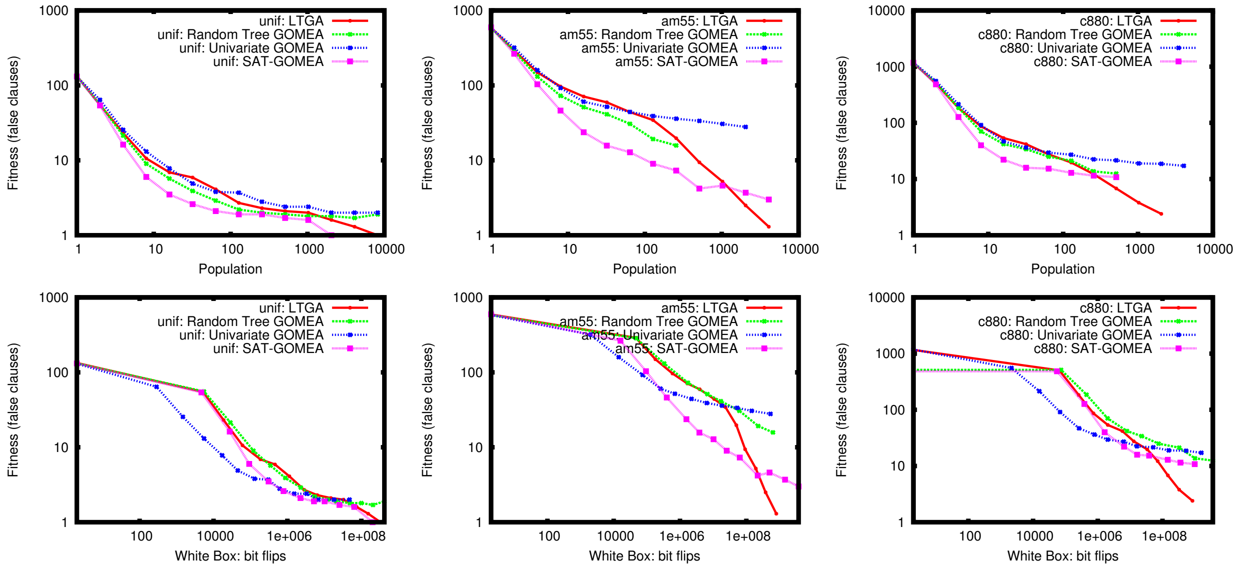

Figure 5: Comparison of performances of GOMEA instances on differently structured benchmarks in the White Box Optimization setting.

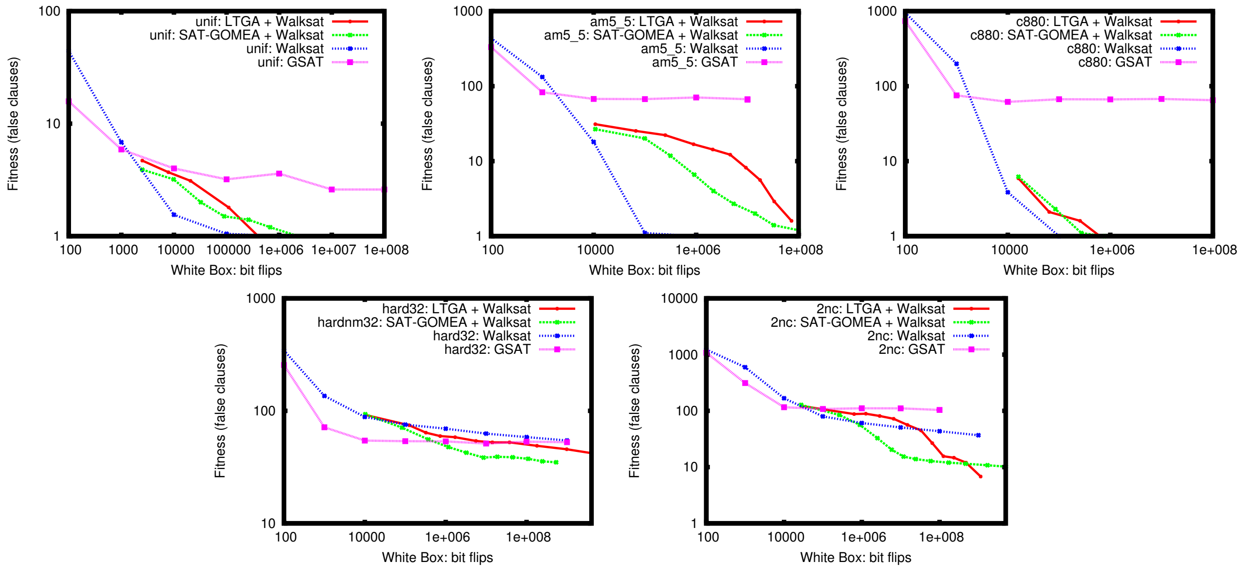

Figure 6: Comparison of performances of the LTGA+Walksat, GOMEA-SAT+Walksat and stand-alone Walksat and GSAT algorithms