The Roles of Local Search, Model Building and Optimal Mixing in Evolutionary Algorithms from a BBO Perspective

Peter A.N. Bosman
Centrum Wiskunde & Informatica (CWI)
P.O. Box 94079
1090 GB Amsterdam
The Netherlands
Peter.Bosman@cwi.nl

Dirk Thierens
Department of Information and Computing Sciences
Utrecht University
3508 TB Utrecht
The Netherlands
Dirk.Thierens@cs.uu.nl

ABSTRACT

The inclusion of local search (LS) techniques in evolutionary algorithms (EAs) is known to be very important in order to obtain competitive results on combinatorial and real-world optimization problems. Often however, an important source of the added value of LS is an understanding of the problem that allows performing a partial evaluation to compute the change in quality after only small changes were made to a solution. This is not possible in a Black-Box Optimization (BBO) setting. Here we take a closer look at the added value of LS when combined with EAs in a BBO setting. Moreover, we consider the interplay with model building, a technique commonly used in Estimation-of-Distribution Algorithms (EDAs) in order to increase robustness by statistically detecting and exploiting regularities in the optimization problem. We find, using two standardized hard BBO problems from EA literature, that LS can play an important role, especially in the interplay with model building in the form of what has become known as substructural LS. However, we also find that optimal mixing (OM), which indicates that operations in a variation operator are directly checked whether they lead to an improvement, is a superior combination of LS and EA.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Experimentation

Keywords

Evolutionary Algorithms, Genetic Algorithms, Estimation-of-Distribution Algorithms, Mixing, Local Search

GECCO'11, July 12–16, 2011, Dublin, Ireland.

1. BACKGROUND, AIM AND SCOPE

1.1 Local Search

In many problems of interest, local features of the search space are highly informative with respect to how to improve a given solution. For this reason, in discrete (combinatorial) spaces, which we focus on here, many Local Search (LS) techniques exist, such as e.g. Hill Climbing (HC) and Tabu Search [3]. LS techniques are well known in the fields of Artificial and Computational Intelligence (AI/CI) as well as in combinatorial optimization. In the former, they are typically applied as Black-Box Optimization (BBO) techniques where no assumptions are made about the problem at hand [16]. Many success stories however also stem from LS techniques that are tailored to a specific (combinatorial) problem [1]. LS techniques are by themselves often already capable of obtaining solutions of good quality and their use is widely accepted in applications and research. LS techniques are particularly fast if only small changes are made and the corresponding change in solution quality can be computed efficiently (i.e. a *partial* evaluation), rather than requiring a complete new evaluation of the optimization function. This is typically possible for combinatorial optimization problems, but clearly not in a BBO setting.

1.2 Evolutionary Algorithms

The scope of Evolutionary Algorithms (EAs) is broader in that the focus is on improving the (average) quality of a set of solutions, called the population, rather than a single solution [4]. An advantage that EAs therefore have is that the collective information stored in the population can be used to generate new solutions. If correlations between the quality of a solution and instances of certain variables exist, multiple solutions will exhibit similar substructures. Detecting and exploiting the existence of such substructures, e.g. via proper mixing, then allows EAs to construct improved solutions in a parallel manner that is not possible in a single-solution LS. Although thereby typically able to reach a higher quality, EAs are also typically slower to arrive at that quality. This difference is largest when LS techniques can use partial evaluations but EAs can't because of large variations made when constructing new solutions. For large instances of combinatorial problems such as traveling salesman instances with thousands of cities, LS techniques can often still provide good-quality solutions within a reasonable amount of time whereas (general) EAs typically can't.

1.3 Evolutionary Local Search

A sensible question is whether the best of both worlds in performance can be obtained by incorporating LS techniques in EAs, i.e. to obtain higher quality solutions than LS alone while being much faster than EA alone. Given the low computational effort of LS techniques, they can be applied to each solution after it is generated in an EA, both initially and as a result of variation operators. The EA then works in the space of local optima that are found by the LS techniques. Such combinations are known under the names of Memetic Algorithms [11] (MA) and Evolutionary or Genetic Local Search [21] (ELS/GLS). Indeed, many studies report better quality than when the LS techniques are used alone, and far smaller computing times than when EAs are used alone. The fact that results obtained by these methods often improve upon the state-of-the-art is testimony to their efficiency. However, much of this efficiency can be contributed to being able to perform partial evaluations and thus this may change when faced with a problem in a BBO setting.

1.4 Model Building

An important key to the success of EAs is that the bias induced by variation matches the structure of the problem. The aforementioned mixing properties of EAs are only beneficial if this leads to the exchange of partial solutions that jointly represent an important contribution to quality. If such information about relations between variables, which is typically called *linkage*, is lost as a result of disruptive mixing, the probability of success can decrease dramatically [17]. It is therefore important to tailor the operators of variation to the problem at hand. In a BBO setting however, this is not possible. To still ensure favorable mixing properties, learning techniques can be employed to statistically analyze the solutions that have been evaluated so far. Typically, learning is used to find the most fitting instance of a predetermined model. The most prominent development along this line are Estimation-of-Distribution Algorithms (EDAs) in which a probabilistic model is learnt (or, synonymously, built) and a corresponding probability distribution is sampled to generate new solutions [10, 12]. The probabilistic models are often built around the concept of statistical dependencies between variables, which corresponds directly to the notion of linkage.

1.5 Local Search and Model Building

1.5.1 Robust Black-Box EA and Problem-Specific LS

EDAs aim to provide a more robust, encoding-independent manner of performing evolutionary search and therefore almost always target a BBO setting. As such, partial evaluations are typically assumed not to be possible in EDA research and hence the lack of speed of an evolutionary search is not considered an issue, but rather is the number of required function evaluations. In that sense, EDAs are ideal candidates for the MA/ELS/GLS format. LS techniques exploit problem-specific knowledge to get to local optima quickly so that the statistical analysis that underlies variation is based on the common characteristics of high-quality solutions only.

1.5.2 Improved Model Building via LS

An interesting effect of combining model building (in EDAs) with LS techniques is that the model building may be improved. For model building, selection in an EA essentially reduces noise [14] because regularities stand out more. However, selection typically isn't a very strong process and unless the search is near termination, the reduction in noise will not be substantial. Increasing the selection intensity only drives up the required population size to ensure sufficient diversity for the multi-generational search, which in turn increases computation time because performing learning on a larger population will require (much) more time. If LS techniques are applied however, starting from different (random) solutions, the noise is strongly reduced while the diversity isn't reduced too much (as long as different local optima are found). Suboptimal instances, that for strongly linked problem variables are a source of noise for the learning task, are effectively weeded out. Indeed, it was recently found that the population size and the number of function evaluations required by an EDA were reduced in a corresponding EDLS [15]. However, the function evaluations for the LS technique were not counted, which is misleading from a BBO perspective. In this paper we shall therefore take a look at performance when also the function evaluations required by LS are counted.

1.5.3 Driving Black-Box LS with Model Building

Another interesting effect of combining model building (in EDAs) with LS techniques has recently gathered more attention in literature. As mentioned above, many models in EDAs identify statistical dependencies, or linkage, between variables. This information can also be exploited in LS. More precisely, if a set of variables is identified as dependent, or linked, LS can be made to search the joint neighborhood of these variables to ensure that positive synergistic effects of changing these variables together are processed. Recent studies of such methods of neighborhood configuration for LS, which is called substructural LS, have shown that an improvement in performance can be obtained [8, 9].

1.6 Optimal Mixing

Very recently, the notion of optimal mixing (OM) was coined [19] in a study that followed up the introduction of the Linkage Tree Genetic Algorithm (LTGA) [18]. In the LTGA, variation is a recombination of two parents. Specifically, recombination is of the crossover type where crossover follows the structure dictated by a linkage tree (LT). A LT is the result of performing a statistical analysis in the form of hierarchical clustering. Importantly, whenever crossover is performed based on a linkage group, a check is carried out to see if the crossover operation was for the better, i.e. whether one of the children is better than both parents. This additional check, which requires additional function evaluations, underlies the notion of OM. OM is not restricted to crossover but can be applied to any variation operator. Instead of fully creating new solutions first and only then evaluating fitness, in OM an existing parent solution is chosen and cloned first. Then, during variation, each time a partial solution is generated, such as is the result of crossing over a part from another parent as reminiscent of GAs or sampling values for a subset of variables as reminiscent of EDAs, this part is used in the full solution that was cloned. If this leads to the solution being improved, the improved solution is kept, otherwise the solution is left unchanged. Variation continues, each time testing whether newly generated partial solutions lead to improvements. Thus, during variation, similar to the

modus operandi of a LS technique, the quality of a solution monotonically improves.

Following the definition of OM, an improvement over the initial LTGA was proposed. The main difference is that instead of using the same parent to perform a single complete OM crossover with, each time a crossover action is to be performed, a new donor parent is randomly selected from the parent pool. If maximum-likelihood estimates are used, this is similar to sampling a probability distribution, which is reminiscent of EDAs rather than GAs, indicating that LTGA could also be claimed to be an EDA. LTGA, especially in its latest version, was found to be highly efficient, outperforming such advanced algorithms as ECGA on GA-difficult optimization problems [19].

1.7 Aim and Scope

OM transforms the more common form of variation into one that is similar to that of LS techniques. One question is therefore whether the effectiveness of LTGA is due to the fact that the role of LS is fully fused with that of variation or whether similar performance can be obtained by performing LS techniques and variation side by side as in MA/ELS/GLS. Another question is what happens if we move to the other side of the spectrum and only apply deterministic LS techniques that try all possibilities in a neighborhood. These techniques can still be driven by similar model-building approaches as used in LTGA (or other EDAs). The LTGA in that sense is in the middle of the spectrum, performing not a full deterministic LS with all possibilities in a learnt neighborhood, but rather a stochastic LS in that neighborhood, attempting to mix only a single partial solution copied from a single parent in with another solution. A further important question is whether LS, both in MA/ELS/GLS format and in the form of OM, helps when faced with a problem in a BBO setting.

2. FEATURED EAS

In the remainder we denote the population size by n and the number of problem variables by l.

2.1 sGA

Arguably the simplest and most common EA, especially for discrete spaces, is the simple Genetic Algorithm (sGA), although no unique definition exists for it. In the GA used here, n new solutions are generated by randomly pairing 2 parent solutions and creating 2 offspring solutions via uniform crossover. In order not to lose good solutions, i.e. to ensure elitism, we combine the current population and the offspring and perform selection on these $2n$ solutions to select n survivors. To ensure convergence by logistic growth of the optimal solution over multiple generations, we use tournament selection with a tournament size of 4.

2.2 ECGA

Although the name suggests otherwise, the Extended Compact Genetic Algorithm (ECGA) [5] can be seen to be an EDA. The probabilistic model that is built is a Marginal Product Model (MPM). A MPM is essentially defined by a partitioning of all variables, i.e. a set of mutually exclusive subsets. The idea is that strongly linked variables can be placed together in marginals so that they are sampled jointly, i.e. together, when generating new solutions.

The method employed in ECGA to configure the MPM is quite a common one in EDA literature. A greedy algorithm is used to optimize a statistical scoring metric, which in this case is the Minimum Description Length (MDL) metric and it should be minimized. The MDL metric defines a trade-off between the quality of the fit of the estimated probability distribution and the number of parameters that is required for the fit. The greedy learning algorithm starts from the univariate structure (all variables are independent) and computes the decrease in MDL metric for all possible merges of two subsets. The merge that results in the biggest decrease in the MDL metric is chosen and the two subsets are replaced by their merged subset. If no merge operation exists that reduces the MDL metric further, the greedy search procedure stops. For more details, see [5, 19].

Complete new solutions are generated by sampling the estimated probability distribution. In the ECGA implementation used in this paper, we generate n solutions in this fashion. Similar to the GA, we then add these solutions to the population and perform tournament selection.

2.3 LTGA and GOMEA

In the MPM any two variables are either fully dependent or fully independent. In the LT any two variables may be dependent according to some subsets, but independent according to others because the LT structure is the result of a hierarchical clustering that is essentially similar to the greedy algorithm used in ECGA. However, unlike for the MPM, when combining subsets, both the combined subset as well as its constituent subsets are in the LT structure. Also different from the learning procedure for MPMs, the combining of subsets continues until only two subsets remain that together contain all problem-variable indices.

In the initial LTGA [18], the distance measure used to select subsets to be combined, is based on the entropy of subsets. For large subsets however, i.e. high up in the linkage tree, the required counting of frequencies of instances is costly. To improve the runtime, average linkage clustering was recently used, which involves only looking at distances between pairs of variables [19].

To use the LT structure, a stack is used upon which first the complete univariate structure is pushed in a random order. Each time two sets are combined, the joined set is also pushed on the stack. When traversing the LT structure, subsets are popped from the stack, meaning that the subsets are considered in reverse merging order. LTGA moreover makes use of optimal mixing (OM). In this paper, we set the population size equal to the number of offspring and let each solution undergo OM. Because each offspring is at least as good as the parent that OM started from, for survivor selection we focus only on the n offspring. To have the same convergence properties as the GA and the ECGA, we can now lower the tournament selection size to 2. Note that the actual selection pressure is much higher as it is partly contained in the OM variation operators.

LTGA is in the class of Genepool Optimal Mixing Evolutionary Algorithms (GOMEA) and has alternatively been identified as LT-GOMEA [19]. EAs in the GOMEA class perform OM using a newly uniformly randomly chosen donor parent for each subset of variables that needs to be crossed over. If the same procedure is used as in ECGA to learn a MPM instead of an LT, the MPM could equally well be combined with OM in GOMEA, leading to MPM-GOMEA.

3. FEATURED LS TECHNIQUES

Because we are taking a BBO perspective, we only consider general LS techniques. Particularly, we focus on Hill Climbing (HC), which is a well-known type of Neighborhood Search (NS). HC considers solutions that are in a neighborhood that is associated with each solution. The most common HC variants are simple Hill Climbing (sHC) and Steepest Ascent Hill Climbing (SAHC). In sHC (also known as first improvement NS) the first improvement found when traversing the neighborhood is accepted. In SAHC (also known as best improvement NS) the entire neighborhood is enumerated and the best improvement is selected. The HC we consider here is actually a combination of both. For a given subset of variables SAHC is performed among the instances that are associated with these variables. However, regarding all available subsets of variables (e.g. the mutually exclusive subsets in a MPM), sHC is performed. In other words, all subsets of variables are considered one after the other. For each subset, the best instance is selected. If that instance is an improvement over the current instance, it is immediately accepted and the next subset is considered.

The types of subsets of variables that can be considered are either univariate (U) (i.e. each variable separately, which, in the case of binary variables corresponds to a "bitflipper"), MPM or LT. The instances that are associated with a subset are either considered in an Exhaustive (E) manner (all possible instances) or in a Genepool (G) manner (only those instances still found in the current population).

HC can either be repeatedly applied to all subsets until convergence onto a local optimum, or only once for each subset (indicated by HC1). The reason for this option is that the last round in HC only serves to check whether no further improvements are possible. In BBO however, such an additional check can be costly.

4. COMBINATIONS OF EA AND LS

We consider running EAs (GA, ECGA, GOMEA) separately or in MA/ELS/GLS format (GLS, ECGLS, GOMELS), meaning that LS is applied to each solution after it is generated, both initially and after variation.

We also consider two algorithms that could arguably be placed equally well within the combinatorial LS community as in the EA community. The first algorithm, Population-based Local Search (PLS) randomly generates n solutions and then applies LS to each solution. Initially always the univariate structure is used because no learning has been done yet. Next, learning is performed and a structure is determined. HC is then applied again to all solutions, but now following the learnt structure. This is repeated until termination is enforced, similar to EAs. The second variant, Population-based Best Only Local Search (PBOLS), is a combination of more traditional single-solution LS and population-based LS. PBOLS starts similar to PLS, but after learning the very first structure, LS following this structure is then applied only to the very best solution, after which PBOLS immediately terminates.

5. EXPERIMENTS

5.1 Optimization problems

We consider two functions, both of which need to be maximized and are treated as BBO problems, i.e. no partial evaluations are possible. The first function is the mutually exclusive, additively decomposable composition of the well-known order-k deceptive trap functions [2]. We consider subfunctions with $k = 5$:

$$f_{\text{Trap5}}(\boldsymbol{x}) = \sum_{i=0}^{(l/k)-1} f^{\text{sub}}_{\text{Trap-k}}\left(\sum_{j=ki}^{ki+k-1} x_j\right)$$

where

$$f^{\text{sub}}_{\text{Trap-k}}(u) = \begin{cases} 1 & \text{if } u = k \\ \frac{k-1-u}{k} & \textit{otherwise} \end{cases}$$

It is commonly known that the EA needs to detect and process the linkage groups pertaining to the deceptive subfunctions in order for optimization to proceed efficiently. Specifically, using univariate structures the minimally required population size and number of function evaluations scales up exponentially [20]. It is clear that for this problem the MPM structure is a perfect fit.

The second function is the nearest-neighbour overlapping, additively decomposable composition of predetermined, but completely random subfunctions of length k, which is essentially a NK-landscape [13]. This type of NK-landscape problem where the overlap between subfunctions is exactly defined rather than randomly defined as in traditional NK-landscapes has the advantage that the optimum can be computed using dynamic programming [13], which is important when we want to use this type of function to determine the minimally required resources for an EA to find the optimum. We consider subfunctions of length 5 and the maximum overlap of 4 (corresponding to a shift of 1 where the next subfunction starts), but without wraparound:

$$f_{\text{NK-S1}}(\boldsymbol{x}) = \sum_{i=0}^{l-k} f^{\text{sub}}_{\text{NK}}\left(\boldsymbol{x}_{(i,i+1,\ldots,i+k-1)}\right)$$

where $f^{\text{sub}}_{\text{NK}}\left(\boldsymbol{x}_{(i,i+1,\ldots,i+k-1)}\right)$ is a pre-determined, but randomly chosen value in $[0; 1]$.

5.2 Setup

For both optimization problems and for problem lengths up to $l = 200$ we have determined the minimally required population size to solve each problem in 99 out of 100 independent runs and computed the corresponding average number of required function evaluations and the average runtime in seconds. For practical reasons, a limit of 10^8 function evaluations was enforced. Termination was also enforced if at the end of a generation all solutions had the same fitness or the best fitness equalled the known optimal value.

5.3 Results

5.3.1 GA and GLS

In Figure 1 the results are shown for the simple GA as well as 4 GLS variants, all of which perform HC only following the univariate structure. The problem sizes are kept small deliberately because it is known that the GA with uniform crossover scales up exponentially on these problems. The additional use of the univariate HC isn't expected to lead to much better results since no multivariate interactions are considered there either. Indeed, with respect to the number of required function evaluations no substantial improvements are observed. On the trap functions GLS even

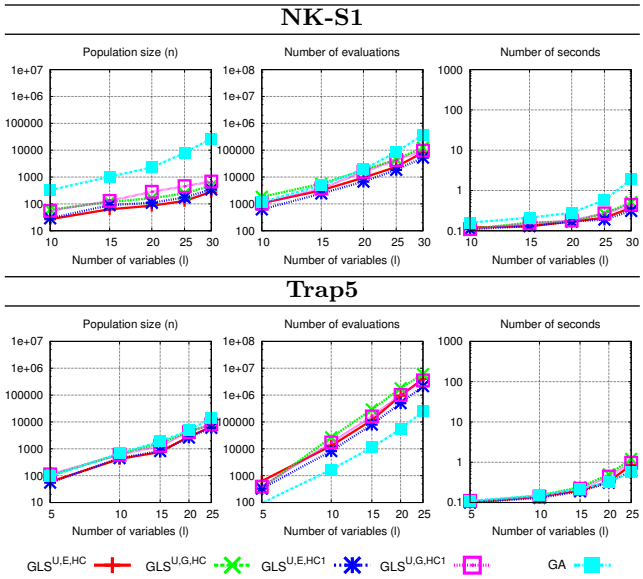

Figure 1: Results for GA and GLS.

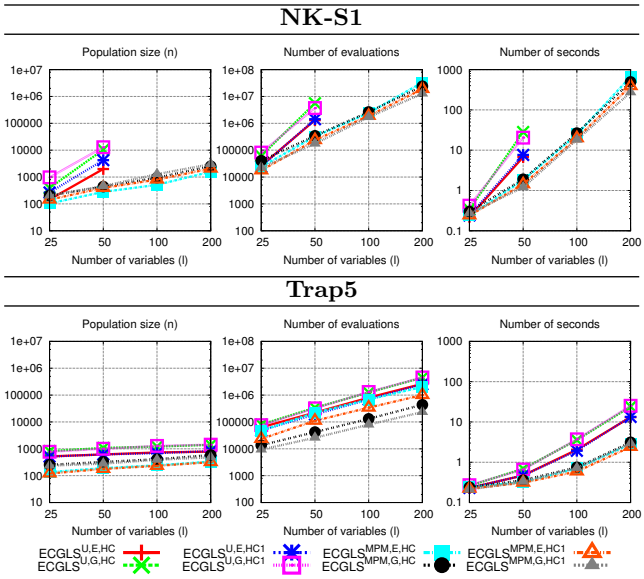

Figure 2: Results for ECGLS.

performs worse. The reason for this is that LS identifies optima and suboptima for individual subfunctions but uniform crossover has only little chance of crossing entire subfunctions over to form better solutions. The probability of generating an optimal building block in one of two offspring is even bigger if both parents are not sequences of local optima. The trap functions however have a rather specific structure and therefore this result shouldn't be readily extrapolated to count for all BBO problem solving. Indeed, when observing the results on the overlapping NK landscapes, the situation is somewhat reversed and the GLS variant with HC1 and exhaustive instance enumeration performs best. The only observable benefit of the integration with LS techniques on both problems is a reduction in required population size, which was only one of the three effects LS integration can have as outlined in Section 1.5. Arguably, model-building techniques are required to be able to observe a bigger impact of the integration of LS techniques as a result of a multiple of the effects as outlined in Section 1.5 coming into play.

5.3.2 *ECGA, ECGLS and GOMEA*

In Figure 2 the effects of using LS both univariately and following the MPM structure on ECGA are shown. First, we disregard the ability to solve the two problems and look only at which variant is best. Similarly to the case of GLS, when using the univariate structure in HC, visiting each variable only once per application, but with an exhaustive instance neighborhood (i.e. a true "bitflipper") is best in terms of minimally required number of evaluations. When following the MPM structure in HC however, better results are obtained if only the genepool instances are used. This is rather straightforward to understand for the trap functions because for each subfunction there is one optimum and one suboptimum and the HC with the correct MPM structure will effectively select the optimal instance without considering all other possible instances. This is however not necessarily directly obvious for the overlapping NK landscapes, but the same strategy performs best. Arguably, the larger the subsets of linked variables, the better the EA by means of selection can already weed out instances that are of no value and even though there may be more interesting candidates than one suboptimum and one optimum as is the case for the trap functions, this number of candidates may still be much smaller than the total number of possible instances, which grows exponentially with the subset size and therefore quickly leads to many more evaluations required by HC.

When considering the ability to solve the problems, Figure 3 shows the best performing ECGLS variants along with ECGA itself. It is known that an EDA performs very well if the probabilistic model can fit well to the structure of the problem. For this reason the EDA can solve the trap functions quite efficiently in terms of the number of required evaluations. Moreover, the effect of noise reduction for the sake of better model building can also be seen as the ECGLS variants have a far smaller minimally required population size than ECGA does. The addition of HC with the univariate structure however does not lead to a reduction in the required number of evaluations.

An EDA will perform worse than a GA with crossover if the probabilistic model can not fit well to the structure of the problem. The performance of ECGA on the overlapping NK landscapes is indeed quite bad. Moreover, the addition of univariate HC does not result in superior scalability, but it does lead to a reduction in the required number of evaluations. Only if HC follows the learnt structure does the scalability improve on the NK landscapes. Even on the trap functions there appears to be a slight improvement. This corroborates the positive influence of using LS to search in a substructural manner as has been reported before in literature relatively recently [8, 9]. But the best contribution is found in the form of optimal mixing (OM). MPM-GOMEA is superior on both problems in terms of function evaluations as well as computing time. On the NK landscapes it requires only a slightly larger population size. This is already testimony that the complete fusion of LS with variation operators in the form of OM is a superior manner of combining EA and LS. Next, we will look more closely at OM and see if the observation of superiority holds when moving to a more advanced form of dependency representation in the form of the LT and further adding LS on top of that.

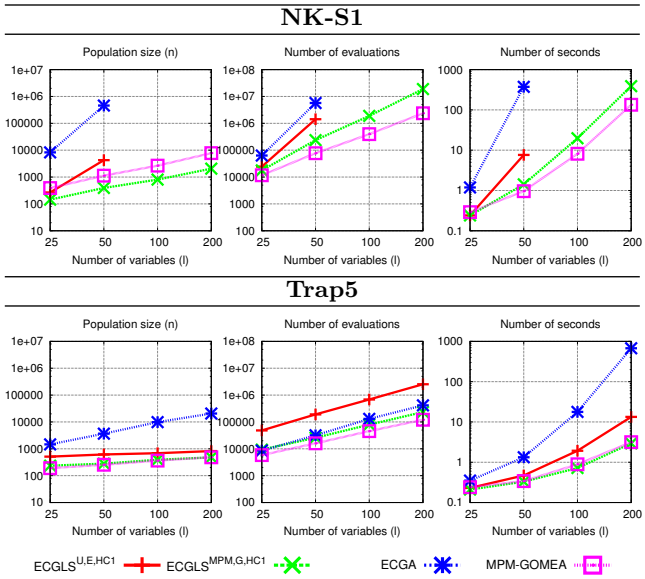

Figure 3: Results for ECGA, ECGLS and GOMEA.

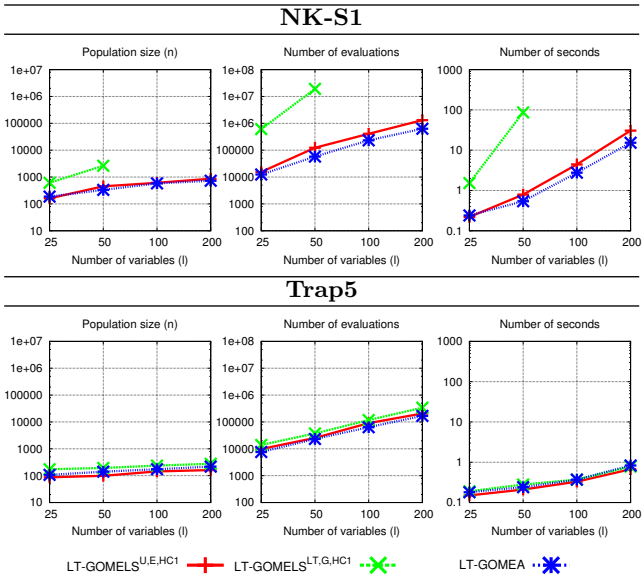

Figure 4: Results for GOMEA and GOMELS.

5.3.3 *GOMEA and GOMELS*

When combining LS with the full LT structure, it is clearly inefficient and computationally intractable to use the exhaustive instance neighborhood. High up in the LT the linkage groups are large and thus the number of instances for these variables is huge. Therefore we only combine the full LT structure with LS by using the genepool of instances. Also, from here on we only show the best variant of HC for each structure. Invariably, for the univariate structure this turns out to be the exhaustive instance neighborhood with a single visit of all variables per application. For the LT structure it is also a *single* visit of the complete linkage tree, for which results are shown in Figure 4.

In terms of required number of evaluations no improvement is obtained over GOMEA when considering the addition of LS in the form of GOMELS. Moreover, when considering the LT structure to search in a substructural manner, different from when considering the MPM structure, the results deteriorate extremely in case of the overlapping NK landscapes. The reason for this is that in order to solve this problem, multiple rounds are required in which the diversity needs to be maintained in the population in order to construct increasingly large parts that also appear in the optimal solution. However, high up in the LT the linkage groups are large; some are at least half the problem size. Considering the genepool of instances then means considering the entire population. Out of these, for such a large group of variables, only one or very few instances will be really good and these will overwrite the corresponding part in almost *every* solution during OM. Consequently, when HC has finished traversing the complete LT, diversity will be dramatically low. To compensate for this overzealous behavior of LS, the population size has to increase enormously, which is contrary to the typically expected effect of adding LS. The number of required evaluations correspondingly sky-rockets because OM itself already requires far more additional evaluations during variation per solution. For the trap functions diversity maintenance over multiple generations isn't really necessary because of the innate separability of the subfunctions. As a result, the addition of LS, even when following the LT structure, isn't highly detrimental to performance. An overall important conclusion however is that LT-GOMEA without additional LS is the superior algorithm for both problems.

Because LS on the complete LT structure is highly costly and dangerous in solving certain problems because of extreme diversity loss, we consider a crossover between LT-GOMEA and LT-GOMELS, which we will indicate by LT-GOM-EA/LS. In LT-GOM-EA/LS an integer k indicates the level from which OM is performed. Any linkage group with more than k variables will be skipped. Complementary, HC only considers the linkage groups in the LT with at most k variables. Because HC for small linkage groups, such as following the univariate structure, was always found to be best when combined with the exhaustive instance neighborhood and HC1, we only tested that particular combination with $k \in \{1, 2, 3, 4, 5\}$. Note that for $k = 0$ LT-GOM-EA/LS is identical to LT-GOMEA, or, identically, LTGA. The results are shown in Figure 5. Although leading to a slight decrease in population size on the trap functions, no significant decrease in required number of evaluations can be observed. Furthermore, on the overlapping NK landscapes no significant reduction in population size could be found and the required number of evaluations even increases slightly. This is a second testimony that the complete fusion of LS with variation operators in the form of OM is a very effective mix of EA and LS and of exploration versus exploitation.

5.3.4 *PLS, PBOLS and GOMEA*

Finally, we consider pushing the fusion of LS and EA further down the spectrum to PLS and PBOLS as explained in Section 4. For both the MPM structure and the LS structure, the best variants for PLS and PBOLS are shown as well as GOMEA. For PLS, in case of the overlapping NK landscapes the HC variant was found to give the best results whereas for the trap functions this turned out to be the HC1 variant. In both cases it was best to determine the instances from the population. For the PBOLS variants, in combination with MPM it worked best to use exhaustive

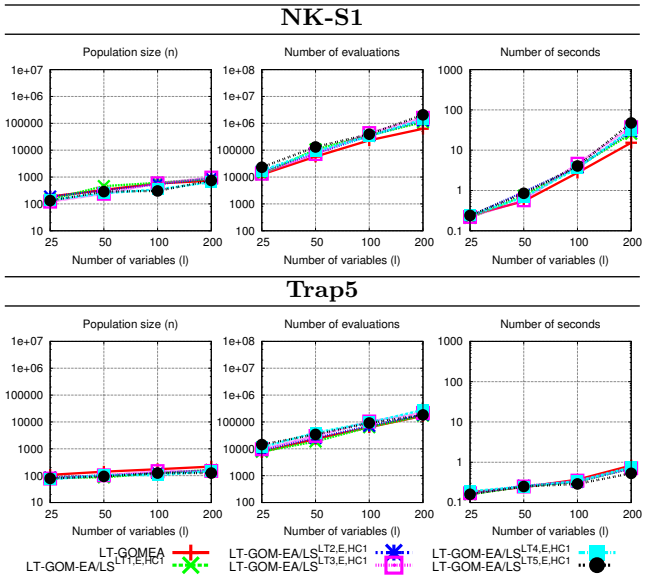

Figure 5: Results for GOMEA and GOM-EA/LS.

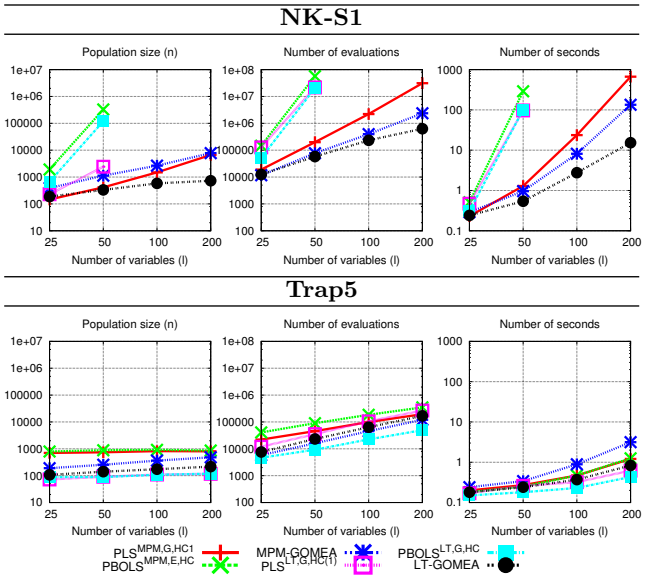

Figure 6: Results for PLS, PBOLS and GOMEA.

instance enumeration but for the LT neighborhood again it was intractable to do so and thus the population was used to determine the instances. It should be noted that for PBOLS the difference between exhaustive and genepool instance enumeration was almost absent on both problems. Results are shown in Figure 6.

It becomes clear again that it may be dangerous to combine the full LT structure with a deterministic LS that considers all possible genepool instances. For the same reason as indicated above, diversity is reduced dramatically. As a result, PLS can't solve the NK landscapes efficiently when combined with the LT structure. When combined with the MPM structure however, diversity loss is substantially less, allowing multiple iterations to be performed, leading to far superior performance on the overlapping NK landscapes.

When observing the results on the overlapping NK landscapes, it becomes clear that with both the MPM structure and the LT structure, PBOLS is not efficient nor scalable because of the large and quickly increasing population size that is required. This is testimony of, even when only LS is considered, the importance of having a population and multiple generations in which important substructures can come to light and grow together when subfunctions are not as perfectly separated as for the trap functions. These results thus underline the value of EAs as an approach in general.

When considering the artificial, but GA-hard and therefore nonetheless important, trap functions however, the most efficient solver is not an EA, but it is PBOLS combined with the LT structure. This indicates that if problems are indeed separable, then LS techniques, albeit population-based for a single generation, are superior to EA methods. However, trap functions don't capture all relevant properties of combinatorial or real-world problems. Considering then that on the overlapping NK landscapes PBOLS and even PLS, which already very closely resembles an EA, are strongly inferior to GOMEA both with the MPM structure and the LT structure and that the gain of PBOLS over GOMEA in case of the trap functions isn't extreme, the strength of EAs combined with LS and model building in the form of OM, even in a BBO setting, can't be ignored.

Finally, considering GOMEA alone, in all aspects of the minimally required population size, the required number of evaluations and computation time, the use of the LT structure is superior over the use of the MPM structure. The LT can be computed efficiently and can further also be used very efficiently in combination with *genepool optimal mixing* by simply univariately randomly selecting new parents for each linkage group in the LT. The resulting LT-GOMEA, or, equivalently, the most recent version of LTGA can therefore be concluded to be a very promising EA for the future.

6. DISCUSSION

This experimental study was done from a BBO perspective. The most important consequence of this is that partial evaluations are not possible. An important question is therefore whether the situation would change in a non-BBO setting such as when considering combinatorial optimization problems. In such problems, when making small changes to a solution, such as moving a node from one set to the other in MAXCUT, the effect on the quality of the solution can be computed in $\mathcal{O}(1)$ time rather than requiring a complete evaluation. For GLS and ECGLS we have seen that they may perform better than, respectively, GA and ECGA on one problem in a BBO setting but worse on another. If partial evaluations were possible however, this could dramatically change. However, GOMEA will likely maintain its relative superior position because OM essentially performs LS operations. Especially when combined with the LT, for the smaller linkage groups partial evaluations are therefore just as well possible, potentially making a specific application of GOMEA equally efficient as MA/ELS/GLS variants.

Regarding the essentially LS operations within GOMEA, it is interesting to consider a third variant of HC that we didn't mention before in this paper, namely Stochastic Hill Climbing (SHC) [7], which can be seen as a specific instance of Stochastic Local Search [6] (SLS). In SHC, the instances for (groups of) variables are not *all* tested like in sHC or SAHC, but rather only one, or a few, possible instances are randomly tested. If we consider testing only a single, ran-

domly chosen instance from only those found in the population, then population-based LS with this SHC variant is *identical* to GOMEA. In other words, GOMEA can validly be identified as a Population-based Stochastic Local Search (PSLS) method, thereby fitting perfectly within the world of LS techniques for combinatorial optimization. Alternatively stated, LTGA, or, equivalently, LT-GOMEA or LT-PSLS, is found at a crossroads between evolutionary computation and combinatorial optimization where the former has been infused with LS inside its variation operators and the latter has been extended with a population and multiple iterations. In either case, key to robustness across a wide range of problems is model building, for which a new powertool is arguably the linkage tree.

7. CONCLUSIONS AND FUTURE WORK

We have investigated the roles of local search, model building and optimal mixing in evolutionary algorithms from a BBO perspective. Optimal mixing can be seen as an integration of local search into the variation operators of an evolutionary algorithm. Such an integration was found to be superior to the more common combinations of local search and evolutionary algorithms that are often indicated as memetic algorithms or evolutionary local search. Moreover, the use of local search was found to be additionally beneficial when model building is used to identify and exploit dependencies between problem variables as is characteristic of estimation-of-distribution algorithms. It helps reduce the population size and noise in the data from which the models are to be built, leading to more precise and computationally faster model building. Moreover, we found that the more recent approach of substructural local search that follows the dependency structure of the built model, is favorable over a univariate local search (i.e. a "bitflipper"). However, optimal mixing was found to be a more efficient manner of exploiting this information still. Furthermore, moving toward the other side of the spectrum where only deterministic local search is performed following the structure of built models in combination with a population was found to be highly inefficient on a problem with overlapping subfunctions but slightly more efficient than optimal mixing on a non-overlapping additively decomposable problem.

All together, we conclude that a population-based stochastic multi-stage process, reminiscent of EAs, is preferable to a single-solution, single- or double-stage process that is reminiscent of more traditional LS techniques. Furthermore, algorithms like LTGA that use of optimal mixing (OM), are in the middle between (traditional) evolutionary computation and (combinatorial) local search and from the results seen so far, that is a very promising spot to be in.

In future work we shall further test the strengths of optimal mixing by considering specific combinatorial optimization problems both in a problem-specific manner as well as from a BBO perspective.

8. REFERENCES

[1] E. Aarts and J. K. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons Inc., New York, New York, 1997.

[2] K. Deb and D. E. Goldberg. Sufficient conditions for arbitrary binary functions. *Annals of Mathematics and Artificial Intelligence*, 10(4):385–408, 1994.

[3] F. Glover and M. Laguna. *Tabu Search*. Kluwer, Norwell, Massachusetts, 1997.

[4] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learing*. Addison Wesley, Reading, Massachusetts, 1989.

[5] G. R. Harik, F. G. Lobo, and K. Sastry. Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ecga). In M. Pelikan et al., editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, pages 39–61. Springer–Verlag, Berlin, 2006.

[6] H.H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, San Francisco, California, 2005.

[7] A. Juels and M. Wattenberg. Stochastic hillclimbing as a baseline method for evaluating genetic algorithms. UC Berkeley technical report CSD-94-834, 1994.

[8] C. F. Lima, M. Pelikan, F. G. Lobo, and D. E. Goldberg. Loopy substructural local search for the Bayesian optimization algorithm. In T. Stützle et al., editors, *Proceedings of the Second International Workshop on Engineering Stochastic Local Search Algorithms — SLS–2009*, pages 61–75, Berlin, 2009. Springer–Verlag.

[9] C. F. Lima, M. Pelikan, K. Sastry, M. Butz, D. E. Goldberg, and F. G. Lobo. Substructural neighborhoods for local search in the Bayesian optimization algorithm. In T. P. Runarsson et al., editors, *Parallel Problem Solving from Nature — PPSN IX*, pages 232–241, Berlin, 2006. Springer–Verlag.

[10] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea. *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms*. Springer–Verlag, Berlin, 2006.

[11] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech concurrent comp. prog. technical report 826, 1989.

[12] M. Pelikan, K. Sastry, and E. Cantú-Paz. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. Springer–Verlag, Berlin, 2006.

[13] M. Pelikan, K. Sastry, D. E. Goldberg, M. V. Butz, and M. Hauschild. Performance of evolutionary algorithms on NK landscapes with nearest neighbor interactions and tunable overlap. In G. Raidl et al., editors, *Proc. of the Genetic and Evol. Comp. Conference — GECCO–2009*, pages 851–858, New York, New York, 2009. ACM Press.

[14] E. Radetic and M. Pelikan. Spurious dependencies and EDA scalability. In J. Branke et al., editors, *Proc. of the Genetic and Evol. Comp. Conference — GECCO–2010*, pages 303–310, New York, New York, 2010. ACM Press.

[15] E. Radetic, M. Pelikan, and D. E. Goldberg. Effects of a deterministic hill climber on hBOA. In G. Raidl et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference — GECCO–2009*, pages 437–444, New York, New York, 2009. ACM Press.

[16] S. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Prentice-Hall Int. Limited, London, 1995.

[17] D. Thierens. Scalability problems of simple genetic algorithms. *Evolutionary Computation*, 7(4):331–352, 1999.

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

[19] D. Thierens and P. A. N. Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference — GECCO–2011*, New York, New York, 2011. ACM Press.

[20] D. Thierens and D. E. Goldberg. Mixing in genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 38–45, San Francisco, California, 1993. Morgan Kaufmann.

[21] N. L. J. Ulder, E. H. L. Aarts, H.-J. Bandelt, P. J. M. van Laarhoven, and E. Pesch. Genetic local search algorithms for the traveling salesman problem. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature — PPSN*, pages 109–116, Berlin, 1991. Springer–Verlag.