A Clustering-Based Model-Building EA for Optimization Problems with Binary and Real-Valued Variables

Krzysztof L. Sadowski
Department of Computer Science
Utrecht University
P.O. Box 80089
3508 TB Utrecht
The Netherlands
k.l.sadowski@uu.nl

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 Computer Science
Utrecht University
P.O. Box 80089
3508 TB Utrecht
The Netherlands
d.thierens@uu.nl

ABSTRACT

We propose a novel clustering-based model-building evolutionary algorithm to tackle optimization problems that have both binary and real-valued variables. The search space is clustered every generation using a distance metric that considers binary and real-valued variables jointly in order to capture and exploit dependencies between variables of different types. After clustering, linkage learning takes place within each cluster to capture and exploit dependencies between variables of the same type. We compare this with a model-building approach that only considers dependencies between variables of the same type. Additionally, since many real-world problems have constraints, we examine the use of different well-known approaches to handling constraints: constraint domination, dynamic penalty and global competitive ranking. We experimentally analyze the performance of the proposed algorithms on various unconstrained problems as well as a selection of well-known MINLP benchmark problems that all have constraints, and compare our results with the Mixed-Integer Evolution Strategy (MIES). We find that our approach to clustering that is aimed at the processing of dependencies between binary and real-valued variables can significantly improve performance in terms of required population size and function evaluations when solving problems that exhibit properties such as multiple optima, strong mixed dependencies and constraints.

CCS Concepts

•**Mathematics of computing** → **Evolutionary algorithms; Mixed discrete-continuous optimization;**

Keywords

Genetic algorithms, Multiple solutions/ Niching, Empirical study

GECCO '15, July 11–15, 2015, Madrid, Spain

 ISBN 978-1-4503-3472-3/15/07…$15.00

DOI: http://dx.doi.org/10.1145/2739480.2754740

1. INTRODUCTION

Research in Evolutionary Computation often focuses on either discrete or continuous optimization. However, many academic and industrial problems are known where discrete and continuous variables are present simultaneously. Additionally, many such problems, especially in the real-world, are accompanied by a set of constraints, which limit the feasibility of the solutions in the search space.

In a black-box setting little is known about the problem landscape. The presence of both discrete and continuous variables in this setting introduces many challenges. Variable dependencies may exist within the discrete and continuous sub-domains, as well as across both domains. Constraints limit the feasibility of the solution space, and multimodality of a landscape may create deceptive attractors towards local optima.

Model-based Evolutionary Algorithms (EAs) have been successful in solving black-box optimization problems for both discrete and continuous domains. EAs can build models by either learning linkage between variables in the discrete domain, or estimating probability distributions in the continuous domain, which leads to identifying search space regions with high quality solutions. The Linkage Tree Genetic Algorithm (LTGA) and Incremental Adapted Maximum Likelihood Gaussian Model Iterated Density Estimation Evolutionary Algorithm (iAMaLGaM)[1] are state-of-the-art model-building EAs for discrete and continuous problems respectively [9] [1]. A recently introduced approach extends the use of these model-based EAs from the individual respective domains to the mixed discrete-continuous domains, by combining and interleaving their model-building mechanisms. Results indicate that integrating those two algorithms can produce an efficient optimization algorithm for various unconstrained mixed discrete-continuous problems [8]. This promising approach is still limited however, as it only considers the continuous and discrete domains independently. By doing so, possible dependencies between discrete and continuous variables are ignored. We aim to improve this approach by adjusting the variable sampling mechanism and by integrating a clustering mechanism to process mixed dependencies.

Clustering the solution space provides the means for focusing the optimization process on multiple important regions

[1]LTGA and iAMaLGaM code is available at http://homepages.cwi.nl/∼bosman/

of the search space simultaneously. This ability has the potential to explore problems with multiple optima efficiently, as well as model and exploit dependencies present in a problem due to existing constraints or inter-variable interactions. We consider different ways of clustering the search space by treating the discrete and continuous variables jointly or independently. Our goal is examining our novel clustering mechanism's ability to improve performance on problems that exhibit dependencies, constraints or consist of multiple optima.

Recent work on Mixed-Integer optimization include Evolution Strategies [6] as well as Bayesian Optimization approaches [5]. We consider different types of constraint-handling mechanisms and compare the results of our clustering based EA with the Mixed-Integer Evolution Strategy (MIES) on constrained problem benchmarks provided in [6].

2. BACKGROUND

The clustering-based model-building EA we introduce in this paper is based on two known model-building Evolutionary Algorithms: LTGA and iAMaLGaM, whose models are carefully interleaved to sample discrete and continuous variables respectively. The performance of the algorithm is tested on unconstrained benchmarks which are designed based on existing discrete and continuous problems, as well as a Mixed-Integer Nonlinear Programming (MINLP) set of constrained problems [4].

2.1 Model-Building EAs

The Linkage Tree Genetic Algorithm (LTGA) is a state-of-the-art model building GA designed for solving discrete problems [9] [10]. Variable dependences are learned in every generation with a hierarchical clustering algorithm and represented in a linkage tree. Every node of this linkage tree is a subset containing between one and $l_d - 1$ problem variables. These nodes represent linkage subsets which are important for the solution. LTGA iterates over all solutions in the population and attempts to improve them: for each solution, the linkage tree is traversed and each subset of the tree is used as a crossover mask between a donor and the parent solution. A random donor from the population is selected for every subset mask. A result of each such crossover is immediately evaluated. If the resulting offspring solution is better or equal than its parent, it replaces the parent. Otherwise, the offspring is discarded.

The Incremental Adapted Maximum-Likelihood Gaussian Model Iterated Density Estimation Evolutionary Algorithm (iAMaLGaM) is a state-of-the-art EDA for real-valued black-box optimization (BBO) [3]. Following the general EDA paradigm, iAMaLGaM estimates a Gaussian distribution every generation from the selected solutions and generates new solutions by sampling the estimated distribution. The mean vector and covariance matrix are estimated incrementally using intergenerational memory decay on maximum-likelihood estimates. A mechanism which scales up the covariance matrix when needed is used to counteract the risk of premature convergence. Finally, the intergenerational Anticipated Mean Shift procedure improves the behavior of iAMaLGaM in slope-like regions of the search space.

2.2 GAMBIT: Integrated LTGA-iAMaLGaM

A novel mechanism for solving unconstrained mixed-integer problems which relies on interleaving the model-building capabilities of LTGA and iAMaLGaM is introduced in [8]. This integrated approach builds models in the discrete and continuous space independently. An important aspect of this mechanism is maintaining a proper balance between sampling the discrete and continuous models. Sampling or updating one of the models too often can lead to premature convergence, or unnecessary exploration of the search space. In standalone iAMaLGaM, each solution is sampled once within a single generation. Standalone LTGA updates each solution up to $2l_d - 1$ times, once for every mask in the linkage tree within a generation. Because of this difference, the individual models cannot be updated at the same time in the integrated version of the two algorithms. To achieve effective sampling rates, every mask evaluation of LTGA for a given solution is followed by sampling the continuous space for the same solution from the iAMaLGaM model. Once all solutions have been sampled the continuous model is updated. The discrete model is only updated after all the masks have been tried on all solutions. This way, the continuous and discrete variables of a given solution are sampled consecutively, however the rebuilding of discrete and continuous models takes place at different times, following the specifications of the individual algorithms.

Sampling of discrete and continuous variables to create new solutions follows mechanisms of LTGA and iAMaLGaM. The continuous model is learned from the top $\tau = 0.35$ fraction of the population, following iAMaLGaM. When continuous variables are sampled, they are accepted without any restrictions. The discrete model is built from the entire population. To generate selection pressure, when a mask is applied, the resulting solution is only accepted if it improves, or is equal to the solution following LTGA.

This integrated LTGA-iAMaLGaM algorithm has not yet been referenced under a specific name. From now on, we will refer to it as the Genetic Algorithm for Model-Based mixed-Integer opTimization, or GAMBIT.

2.3 MIES

In addition to analyzing the performance of our novel clustering-based EA, we take a closer look at results acquired with the Mixed-Integer Evolution Strategy (MIES). MIES extends $(\mu+\lambda)$-ES for continuous problems to mixed-integer search spaces. After a uniformly distributed initial population is created, new solutions are generated by the following procedure. Two solutions are randomly chosen from the current population $P(t)$ to act as parent solutions. A recombination operator is applied to parent solutions, followed by a mutation operator. Those operators are defined differently for continuous, nominal discrete and integer problem variables. This procedure repeats until λ offspring solutions are created. The best μ solutions from the union of the μ parent solutions and the λ offspring are selected and carried over into the population $P(t+1)$ [6].

3. BENCHMARK PROBLEMS

To study the behavior of our algorithm we first consider a set of unconstrained problems. In these problems constraint-handling mechanisms are not needed, which helps in acquiring better insight and understanding of the algorithm behavior. We designed mixed-integer benchmarks based on well-known discrete and continuous problems, which exhibit different problem characteristics. A specification of the in-

dividual functions can be found in Table 1. Table 2 summarizes how these functions are used together to create unconstrained mixed discrete-continuous benchmarks. Initialization range for all the unconstrained problems is [-5,5].

The first mixed benchmark F_1 is the Onemax-Sphere function. It is a concatenation of two problems where all the variables are completely independent. DT5-R.Ellipse, or F_2, is a problem where discrete variables are strongly dependent on each other because of the Deceptive Trap Function behavior, while the continuous variables are strongly dependent on each other due to parameter rotation of 45 degrees. However, no dependence between the discrete and continuous domains exist. Cross-Dependent function F_3 includes dependencies between both domains as well as mixed dependencies between the variables. It is a specific combination of the previously defined F_{DT5} function with the rotated ellipsoid. It is additively decomposable and consists of subfunctions pertaining to blocks of k discrete and k continuous variables. For a trap function with $k = 5$, there are $2^k = 32$ different binary combinations per block. A differently translated and rotated ellipsoid function corresponds with each of those combinations. The continuous function which is being optimized depends on the binary counterpart, introducing dependencies between the discrete and continuous variables that pertain to the same subset. In the function definition, D_i^{block} is a block of five discrete variables. C^{block} are the corresponding five real-valued variables. The D block variables determine which of the 2^k different rotated ellipsoid functions are being optimized, while C block provides the values of the ellipsoid function variables. In this benchmark the number of discrete variables is the same as the continuous variables: $l_d = l_c = l/2$. Finally, Twomax-Cos, or F_4, is a symmetry-breaking problem with three local optima and one global optimum. This function is a combination of the $cosine(\boldsymbol{x_c})$ function with the $Twomax(\boldsymbol{x_d})$ function. There are four symmetric optima - two from the cosine function and two from Twomax, however only one is the global optimum.

Table 1: Continuous and Discrete functions used to define our unconstrained mixed benchmarks

Functions
$F_{Sphere}(\boldsymbol{x_c}) = \sum_{i=0}^{l_c-1} c_i^2$
$F_{R.Ellip.}(\boldsymbol{x_c}) = F_{Ellip.}(\mathbf{R} * \boldsymbol{x_c})$, where $F_{Ellip.}(\boldsymbol{x_c}) = \sum_{i=0}^{l_c-1} 10^{6*i/(l_c-1)} * c_i^2$
$F_{Onemax}(\boldsymbol{x_d}) = \sum_{i=0}^{l_d-1} d_i$
$F_{DT5}(\boldsymbol{x_d}) = \sum_{i=0}^{l_d/k-1} f_{Trap-k}^{sub}(\sum_{j=ki}^{ki+k-1} d_j)$, where $f_{Trap-k}^{sub}(u) = \begin{cases} 0 & : if\ u = k \\ 1-(k-1-u)/k & : otherwise \end{cases}$
$F_{Twomax}(\boldsymbol{x_d}) = max(F_{Onemax}, F_{Zeromax})$

Table 2: Unconstrained Problems

ID	Definition
F_1	$F_1(\boldsymbol{x_d}, \boldsymbol{x_c}) = F_{Onemax}(\boldsymbol{x_d}) + F_{Sphere}(\boldsymbol{x_c})$
F_2	$F_2(\boldsymbol{x_d}, \boldsymbol{x_c}) = F_{DT5}(\boldsymbol{x_d}) + F_{R.Ellip.}(\boldsymbol{x_c})$
F_3	$F_3(\boldsymbol{x_d}, \boldsymbol{x_c}) = \sum_{i=0}^{0.5l/k-1}(1 + 10^a f_{sub}^{trap}(\sum_{j=ki}^{ki+k-1} d_j)) * (1 + f_{Ellipse}^{sub}(D_i^{block}, C_i^{block}))$
F_4	$F_4(\boldsymbol{x_d}, \boldsymbol{x_c}) = F_{Twomax}(\boldsymbol{x_d}) + F_{COS}(\boldsymbol{x_c})$

Many industrial mixed continuous-discrete problems include constraints. Very often, it is the constraints, not the function itself that makes the problem much more difficult to solve. We consider a set of Mixed-Integer problems from the MINLP Benchmark Library [4]. Specification of the objective function, constraints and parameters ranges can be found in Table 3. Minimization is assumed for both unconstrained and constrained problems.

4. CLUSTERING-BASED MIXED DEPENDENCY MODELING

4.1 Motivation

Understanding the underlying problem structure can be key in effectively optimizing a black-box mixed-integer problem. GAMBIT is a promising recent approach which joins the model building capabilities of LTGA and iAMaLGaM. Variable dependencies are thereby still independently modeled for each domain, while the frequency of sampling solutions and model updates is carefully balanced [8]. Considering dependencies between different types of variables (i.e. discrete and continuous) could be a very important step in solving difficult problems more efficiently. We study if integrating a clustering-based approach with GAMBIT improves performance on problems that exhibit variable dependencies, constraints or contain multiple optima.

Similar to LTGA, one could arguably consider explicitly modeling mixed dependencies on a variable level by means of calculating mutual information between discrete and continuous variables. This information could then be used to learn important mixed linkage subsets in a similar fashion to how LTGA generates and samples from a linkage tree of discrete subsets. To process dependencies for such mixed masks, clustering could be used, for instance to model the distribution in the continuous space conditionally on a clustering of the discrete space. However, mixed mask configurations acquired this way may differ in subsequent generations, resulting in intrinsically different ways of clustering the search space and learning the parameters of the Gaussian distributions in iAMaLGaM. This invalidates the use of the intergenerational mechanisms of iAMaLGaM as they are currently defined. It is not straightforward to change these mechanisms accordingly.

This does not completely invalidate the use of clustering, however. Clustering on the solution level allows the preservation of these crucial intergenerational mechanism, as models created in the previous generation can be mapped to models resulting from the clustering in the current generation. Furthermore, clustering has the capacity to partition the search space into regions representing different optima if they exist within a problem space. This feature is likely to also improve performance of GAMBIT and further motivates the use of a cluster-based mechanism.

4.2 Implementation

The overview of this clustering-based mechanism is as follows. A constant number k of clusters is predetermined. A population is generated randomly with n solutions that are evaluated. From this population k equally-sized (n/k) clusters which exert some level of similarity in accordance with the chosen distance metric are selected. Details on how the clusters are determined is explained later in this section. It is possible for the same solution to exist in multiple

Table 3: Specifications of the MINLP selected constrained benchmark problems

Name	Function	Constrains	Range
F_5	$2r_1 + d_1$	$-r_1^2 - d_1^2 \leq -1.5, r_1 + d_1 \leq 1.6$	$r_1 \in [0, 1.6]$ $d_1 \in \{0, 1\}$
F_6	$2r_1 + 3r_2 + 1.5d_1 + 2d_2 - 0.5d_3$	$r_1^2 + d_1 = 1.25, r_2^{1.5} + 1.5d_2 = 3$ $r_1 + d_1 \leq 1.6, 1.333r_2 + d_2 \leq 3$ $-d_1 - d_2 + d_3 \leq 0$	$r_{1,2} \in [0, 10]$ $d_{1,2,3} \in \{0, 1\}$
F_7	$0.8 + 5(r_1 - 0.5)^2 - 0.7d_1$	$-exp(r_1 - 0.2) - r_2 \leq 0$ $r_2 + 1.1d_1 \leq -1, r_1 - 1.2d_1 \leq 0$	$r_1 \in [0.2, 1]$ $r_2 \in [0.22554, -1]$ $d_1 \in \{0, 1\}$
F_8	$6 + (r_1 - 1)^2 + (r_2 - 2)^2$ $+(r_3 - 3)^2 - d_1 - 3d_2 - d_3$ $-0.693147180559945d_4$	$r_1 + r_2 + r_3 + d_1 + d_2 + d_3 \leq 5$ $r_1^2 + r_2^2 + r_3^2 + d_3 \leq 5.5$ $r_1 + d_1 \leq 1.2, r_2 + d_2 \leq 1.8$ $r_3 + d_3 \leq 2.5, r_1 + d_4 \leq 1.2$ $r_2^2 + d_2 \leq 1.64, r_3^2 + d_3 \leq 4.25$ $r_3^2 + d_2 \leq 4.64$	$r_{1,2,3} \in [0, 10]$ $d_{1,2,3,4} \in \{0, 1\}$
F_9	$-5r_1 + 3r_2$	$8r_1 - 2r_1^{0.5}r_2 + 11r_2 + 2r_2^2 - 2r_2^{0.5} \leq 39$ $r_1 - r_2 \leq 3, 3r_1 + 2r_2 \leq 24$ $r_2 - d_1 - 2d_2 - 4d_3 = 1, d_2 + d_3 \leq 1$	$r_1 \in [1, 10]$ $r_1 \in [1, 6]$ $d_{1,2,3} \in \{0, 1\}$
F_{10}	$(r_4 - 1)^2 + (r_5 - 2)^2 + (r_6 - 1)^2$ $-log(1 + r_7) + (r_1 - 1)^2$ $+(r_2 - 2)^2 + (r_3 - 3)^2$	$r_1 + r_2 + r_3 + d_1 + d_2 + d_3 \leq 5$ $r_6^2 + r_1^2 + r_2^2 + r_3^2 \leq 5.5, r_1 + d_1 \leq 1.2$ $r_2 + d_2 \leq 1.8, r_3 + d_3 \leq 2.5, r_1 + d_4 \leq 1.2$ $r_5^2 + r_2^2 \leq 1.64, r_6^2 + r_3^2 \leq 4.25$ $r_5^2 + r_3^2 \leq 4.64$ $r_4 - d_1 = 0, r_5 - d_2 = 0$ $r_6 - d_3 = 0, r_7 - d_4 = 0$	$r_{1,2,3} \in [1, 10]$ $r_{4,5,6,7} \in [0, 1]$ $d_{1,2,3,4} \in \{0, 1\}$

clusters. A separate continuous-discrete model is associated with one cluster of solutions. Within each cluster, the variable linkages and Gaussian parameters for the discrete and continuous variables are learned respectively. Based on the learned information, each model instance generates n/k new solutions, which are added into the main population. The clustering process begins anew for every generation. After clustering takes place, the model instances from previous generation are matched with the newly created clusters using a well-known distance-minimizing Hungarian method, where the cluster distance is the sum of all pairwise distances between solutions. The distance between individual solutions is the same as the one used for clustering. By doing so, the intergenerational mechanisms needed by iAMaLGaM are preserved. The clustering process is shown in Figure 1.

The clustering process itself takes place at the beginning of every generation. Depending on the number of clusters k, cluster leaders are selected. The best solution available in the population is always selected as a cluster center. Other centers are solutions with the furthest distance from all already chosen centers. Once the centers are determined, n/k solutions closest to the chosen centers are added to the clusters in accordance with the chosen distance metric. This cluster selection process is based on a similar approach used for Multi-Objective optimization in [2].

```
Clustering-Based Model-Building EA
for i ∈ {0, 1, ..., n − 1} do
    P_i ← CreateRandomSolution()
    EvaluateFitness(P_i)

while ¬TerminationCriterionSatisfied do
    C ← PopulateClusters(P)
    for j ∈ {0, 1, ..., C} do
        P_i ← IntegratedAlgorithm(C_i)
    C ← MatchClusters
```

Figure 1: Pseudo-Code overview of the Clustering-based mechanism

4.3 Distance Measures

By clustering the current solution space we attempt to identify groups of solutions within the entire population that represent common properties or dependencies. In order to perform such clustering, a distance measure is required. We want to compare the performance of our algorithm when the clustering process takes into account discrete and continuous variables together and when only one sub-domain is used. To accomplish this we consider four different distance measures. Two of them perform clustering using a distance measure which considers the discrete and continuous domains independently, while the remaining two attempt to process the discrete and real-valued domains jointly.

Independent distance measures consider either the discrete or continuous domain only. For the discrete measure, clustering is done only on the discrete variables with the use of the Hamming distance between two solutions. In the continuous variant, Euclidean distance is calculated.

In order to consider both discrete and continuous variables together, a joint distance metric is required. One option is to consider all variables as discrete. For this, the continuous domain is discretized. Each continuous variable is converted into a binary one, depending on if it lies above or below the median value of the current population for this variable. This is a very crude discretization, as each continuous variable is represented only by one bit. Once all variables are processed, the distance between different solutions is calculated using the Hamming distance which considers the

```
Integrated Model Sampling
for i ∈ {0, 1, ..., n − 1} do
    P_i ← CreateRandomSolution()
    EvaluateFitness(P_i)

while ¬TerminationCriterionSatisfied do
    LearnDiscreteModel(P)
    for i ∈ {0, 1, ..., (2l_d − 1 + 2l_c − 1)} do
        if sampleType(i) = continuous then
            S ← TruncationSelection(P, τ)
            LearnContinuousModel(S)
            for j ∈ {0, 1, ..., n − 1} do
                X_i ← GenerateContinuousPart(P_i)
        else
            for j ∈ {0, 1, ..., n − 1} do
                X_i ← GenerateDiscretePart(j, X_i, P)
    P ← X

GenerateContinuousPart(P_i)
    X_c ← SampleContinuousModel()
    X ← X_c ∪ P_{i_d}
    EvaluateFitness(X)
    return X

GenerateDiscretePart(j, X_i, P)
    X_prev ← X_i
    donor ← GetRandomSol(P)
    X_d ← CopySubset (j, donor, X_i)
    X ← X_{i_c} ∪ X_d
    EvaluateFitness(X)
    if fitness(X) ≥ fitness(X_prev) then
        return X
    else
        return X_prev
```

Figure 2: Pseudo-code for generating solutions with GAMBITs Learning Models.

discretized continuous variables and the discrete variables together. Encoding continuous variables with more than one bit would unbalance calculating the Hamming distance, giving the continuous variables more weight.

Another way of modeling mixed similarities is to consider discrete and continuous variables together in the continuous domain. The continuous variables are normalized and scaled down to a 0-1 range where the current largest value in the population takes on the value of one, and the smallest zero. Binary variables are then treated as real values. This way Euclidean distances can be computed between two solutions using discrete and continuous variables together.

4.4 Generating Offspring

In the first generation k continuous-discrete models are created. Each cluster of solutions is assigned to one model. In this phase, each model uses solutions provided by the clustering mechanism to learn variable dependencies and update the discrete and continuous models. With this new information, new solutions are generated from every model. We make a significant change to the sampling mechanism of the continuous-discrete model. The algorithm proposed in [8] achieves a good balance between sampling the discrete and continuous models by sampling the entire continuous space every time a discrete mask is sampled. The downside of this approach is that the rate of sampling the continuous space relies on the number of discrete masks only. This means that continuous evaluations will happen every time discrete ones do. This is not necessarily desired behavior, as we might want to increase the number of times continuous domain is sampled when the ratio of continuous to discrete variables is large, and lower it when this ratio is small. We remedy this by including the number of continuous variables as a factor in determining how many continuous evaluations will take place every generation. More specifically, the number of sampled discrete masks is $2l_d - 1$, as specified by LTGA. Instead of basing the continuous sampling on this number as well, we now sample the continuous space $2l_c - 1$ times. Additionally, instead of a discrete sampling always preceding a continuous one, sampling happens in a randomized order. These changes allow GAMBIT to allocate additional processing to continuous variables when they make up a large portion of the total problem size, and saves on unnecessary evaluations in the continuous space when real-valued variables make up a small part of the problem. Integrated model building and solution generation done by GAMBIT is summarized in Figure 2.

In experiments on constrained problems we consider different constraint-handling techniques with our clustering-based algorithm. Constraint violation checking takes place every time a new solution is introduced to the population.

5. CONSTRAINT HANDLING

A constrained mixed discrete-continuous problems can be defined as follows:

$$min\ f(\boldsymbol{x_d}, \boldsymbol{x_c})$$

$$\text{s.t. } \mathbf{h}(\boldsymbol{x_d}, \boldsymbol{x_c}) = 0, \quad \mathbf{g}(\boldsymbol{x_d}, \boldsymbol{x_c}) \leq 0$$

Where x represents the solution

$$x = \boldsymbol{x_d}\boldsymbol{x_c} = d_0...d_{l_d-1}\ c_0...c_{l_c-1}$$

where $d_i \in \{0, 1\}, c_i \in \mathbb{R}$ and $\boldsymbol{x_d}$, $\boldsymbol{x_c}$ are the sets of all discrete and continuous variables, respectively. f is the objective function. $\mathbf{h}$ and $\mathbf{g}$ are the sets of equality and inequality constraint functions respectively.

Mixed discrete-continuous problems are often accompanied by constraints. No matter how good the objective function value of a certain solution is, if any of the constraints is violated, the solution is considered infeasible. A set of equality and inequality constraints may exist simultaneously within a problem. Within a black-box setting, we cannot make assumptions about the number or type of constraints that may be present. The only feedback we may hope to have is an indication of how much the constraints are violated. We obtain this indicator by computing a squared sum of all constraint violations for a given solution. This constraint violation value indicates the relative strength of constraint violation. A constraint violation value of 0 indicates a feasible solution. Because of precision issues, in our experiments a solution with a constraint value less than 10^{-10} is considered feasible.

We consider three prominent constraint handling techniques: Constraint Domination, Global Competitive Ranking and the Dynamic Penalty Function [7]. Constraint domination strongly favors feasible solutions. A feasible solutions is always better than an infeasible one. A better function value determines which solutions are better when solutions are feasible. Infeasible solutions are ranked by their constraint violation value. Constraint Domination can

quickly converge on high function value solutions in the feasible space, but avoids exploration of infeasible regions when possible.

Global competitive ranking maintains two independent rankings. All solutions (feasible and unfeasible) are ordered by their objective function as well as their constraint violation values independently. The final ranking is generated based on a combination of the objective function and constraint violation rankings. This approach finds a balance between solutions in the feasible and infeasible space. Infeasible solutions of high objective function values are kept in the population, which could potentially allow for good feasible regions nearby them to be explored.

We also consider the Dynamic Penalty Method for constraint handling. In this method the fitness value of a solution is diminished by a penalty factor if the solution is not feasible. The penalty factor is proportional to the constraint violation value. Additionally, this penalty factor is proportional to the number of generations which already passed. If a solution cannot reach feasible values, over time the penalty factor becomes stronger. With this method promising infeasible regions can be explored, however as time passes, feasible solutions become increasingly favorable. In our implementation the penalty function value is the square of the total constraint violation value multiplied by the number of generations which had already passed.

6. RESULTS

Experiments are performed on a set of unconstrained as well as constrained benchmarks. Bisection was performed for all problems to determine the population size for which the number of evaluations is minimal, while the optimum is reached at least 19 out of 20 runs. The maximum population size used was 2500.

6.1 Unconstrained Problems

As specified in the previous section we consider two approaches that cluster the population using only information from one domain (either discrete or continuous domain) or from both domains jointly. Figure 3 shows representative results of our experiments with different clustering distance measures. "Discrete" and "Continuous" labels refer to the distance measures considering only the discrete and real-valued domains independently. "Mixed (D)" and "Mixed (C)" refer to the clustering mechanism using joint information, by discretization of the continuous variables, or by treating discrete variables as continuous respectively. The results are presented on the F_4 benchmark that exhibits symmetries or large, prominent multi-modalities with a cluster setting $k = 2$. The presented results are representative of the behavior with larger cluster sizes.

From 3 we can see that the "Mixed" configurations perform significantly better in terms of minimally required population size to solve the problem, as well the number of evaluations. This indicates that by clustering on the solution level when considering both discrete and continuous variables together, the clustering mechanism is capable of capturing some mixed dependencies, which allow GAMBIT to exploit the search space more efficiently.

Out of the two configurations which consider mixed variable dependencies, "Mixed (C)" performs slightly better. This could be attributed to the fact that discretizing continuous variables with "Mixed (D)" may result in significant loss

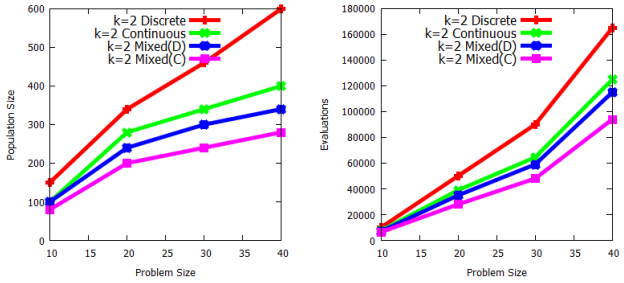

Figure 3: Effects of clustering on Discrete and Continuous variables independently vs. jointly on F_4

of information in comparison with the "Mixed (C) " configuration. This loss of information effect is not as strong with "Mixed (C)" where the continuous variables are normalized instead.

Further experiments on the unconstrained benchmarks are conducted with different cluster sizes k and summarized in Figure 4. In all graphs in this figure the mixed distance measure is used. "NoCluster" refers to the original integrated algorithms as presented in [8]. "$k = 1$" represents our new implementation of the algorithm with cluster size of 1. The results show that our new implementation with k=1 is always more efficient than the old approach. The results presented show problem instances with the same number of continuous and discrete variables. The "k=1" configuration was also tested on functions with $f = 0.25$ and $f = 0.75$ fraction of continuous variables and still always outperformed the "NoCluster" configuration.

When k is increased, the behavior changes depending on the type of the benchmark. F_1 is the simplest benchmark in the set. All variables are completely independent, and the search space is not deceptive. A single cluster has no difficulty solving this problem efficiently. F_2 contains strong dependencies in the continuous and discrete domains, but no cross-dependencies. Here a single cluster configuration is also favorable, as the underlying models of GAMBIT are very efficient in dealing with dependencies contained within each domain, without any need to partition the space and process mixed dependencies. Because of this, larger clusters also solve these problems, but are less efficient. The results change when the benchmarks become more difficult. F_3 consists of cross-dependencies in addition to discrete and continuous dependencies. Allowing the clustering mechanism to process the mixed dependencies results in significantly better performance in terms of population size and evaluations needed with $k = 2$ than with $k = 1$. F_4 is a symmetry-breaking problem with multiple optima. Thanks to the processing and clustering the solution space $k = 2$ and $k = 4$ perform much better than a non-clustering approach.

6.2 Constrained Problems

We have tested a selection of MINLP Benchmarks with the Clustering-based Model-Building EA corresponding with the benchmarks used in [6]. Results are presented for two settings of the clustering size: $k = 1$ and $k = 10$. Similarly to the unconstrained problem analysis, we consider different approaches to clustering : "Discrete" and "Continuous" cluster the population based on the continuous and discrete variables respectively only. "Mixed (D)" and "Mixed (C)"

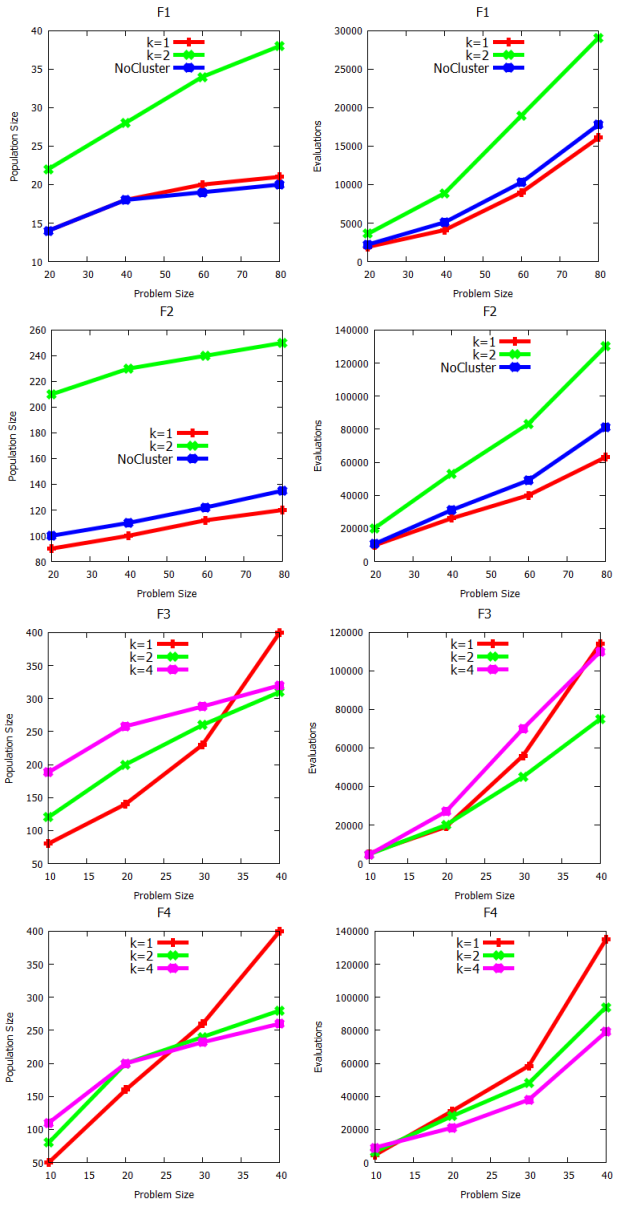

Figure 4: Performance of the clustering mechanism with changing cluster sizes on unconstrained functions

use the discrete and continuous variables together (either in Discrete space or Continuous space). In order to handle constraints, we test each algorithm configuration with different constraint handling techniques: constraint domination (CD), global competitive ranking (GCR) and dynamic penalty function (DPF). Success criteria is defined as reaching the optimal value to a precision of 10^{-5}.

Table 4 summarizes the experiments. It shows that the only consistently successful configurations are Mixed with k=10. This confirms our previous observations and further implies that the clustering algorithm which uses both discrete and continuous variables on which the clustering is performed, is able to successfully process certain dependencies and features of the constrained landscape. For simpler problems, such as F_6, additional clustering is not needed, and a configuration with $k = 1$ can successfully find the optimum. However, in many problems with stronger dependencies and constraints $k = 1$ is not sufficient. Similarly to results from the previous section, "Mixed (C)" outperforms the "Mixed (D)" configuration.

Additionally we observe that constraint-handling techniques have a strong effect on the performance of any configuration of the clustering algorithm. While constraint domination and global competitive ranking perform similarly, the dynamic penalty method is more successful. Performance of MIES on the same benchmarks set is presented in [7] and summarized here in Table 5 which shows the median evaluations needed to reach near-optimal values, as well as the standard deviation of the results. Direct comparison is difficult, as the reported MIES performance does not indicate the success rate. Additionally, our implementation considers different population sizes while MIES does not. MIES however allows for parameterization of its constraint handling method, while we do not. Despite these differences, the results indicate that the clustering algorithm is competitive with MIES in terms of evaluations needed to reach the optimum on most of the tested benchmarks.

Table 5: Median number of evaluations and S.Dev. of MIES

Function	Evals	SD
F_5	7000	$1.9870 * 10^{-13}$
F_6	21700	$3.5616 * 10^{-15}$
F_7	22400	$3.6293 * 10^{-15}$
F_8	23800	$1.5132 * 10^{-13}$
F_9	20300	$2.3570 * 10^{-1}$
F_{10}	131600	$6.0510 * 10^{-1}$

7. CONCLUSIONS AND SUMMARY

In this paper we studied the added value of clustering in a previously introduced model-building EA named the Genetic Algorithm for Model-Based mixed-Integer opTimization (GAMBIT). Using a distance metric that considers discrete and continuous variables jointly, this clustering mechanism partitions the search space based on mixed continuous-discrete variables on the solution level, and learns discrete variable linkage and continuous covariance independently within each cluster. Our results showed that through integration of the clustering mechanism and a more balanced integration of sampling discrete and contentious variables, we significantly improve the performance of GAMBIT for problems exhibiting features such as variable dependencies or multiple optima in a black-box environment. Analysis of performance on unconstrained benchmarks showed that if such features exist in the problem, generating clusters with the consideration of both variable types together performs significantly better than considering the continuous and discrete domains only independently, or not clustering in the solution space at all.

Some drawbacks still exist. This clustering procedure is sensitive to the number of clusters k. If the problem does not exert strong mixed dependencies or multi modality, a high setting for k may lead to unnecessary overhead in terms of minimal required population sizes and number of evaluations.

Table 4: Performance of different variants of the clustering-based algorithms with different constraint handling techniques. Best results are bold. Equal statistical significance (T-test) is annotated with *.

				CD			GCR			DPF	
Problem	**k**	**Clustering**	**Pop**	**Evals**	**Success**	**Pop**	**Evals**	**Success**	**Pop**	**Evals**	**Success**
F_5	1	-	40	1104	1	40	1104	1	**40**	**897***	**1**
F_5	10	Discrete	100	1271	1	100	1271	1	200	970	1
F_5	10	Real	100	1473	1	100	1473	1	100	1402	1
F_5	10	Mixed (D)	100	1350	1	100	1350	1	100	1101	1
f_5	10	Mixed (C)	100	1112	1	100	1112	1	100	910*	1
F_6	1	-	2500	-	0	2500	-	0	2500	0	0
F_6	10	Discrete	480	19014	1	240	19014	1	240	6308	1
F_6	10	Real	2500	39014	0.1	2500	-	0	2500	45665	0.3
F_6	10	Mixed (D)	160	7402	1	180	7812	1	140	6304	1
F_6	10	Mixed (C)	160	6867	1	180	7114	1	**120**	**4001**	**1**
F_7	1	-	70	11627	0.95	70	11104	1	300	30886	1
F_7	10	Discrete	**100**	**3551**	**1**	100	4211	1	400	20422	1
F_7	10	Real	100	6788	1	100	6537	1	500	28995	1
F_7	10	Mixed (D)	100	5330	1	100	6151	1	400	18664	1
F_7	10	Mixed (C)	100	7133	1	100	7715	1	400	19910	1
F_8	1	-	2500	-	0	2500	-	0	2500	-	0
F_8	10	Discrete	1000	146332	0.95	1100	167554	1	900	50885	0.95
F_8	10	Real	2500	-	0	2500	-	0	2500	-	0
F_8	10	Mixed (D)	900	79664	0.95	900	85244	1	600	56744	1
F_8	10	Mixed (C)	800	49774	1	800	52274	1	**400**	**20155**	**1**
F_9	1	-	60	4919	1	70	5067	1	**40**	**2996***	**1**
F_9	10	Discrete	100	6446	1	100	6223	1	100	4359	1
F_9	10	Real	100	6674	1	100	6530	1	100	4102	1
F_9	10	Mixed (D)	100	7350	1	100	7004	1	100	6412	1
F_9	10	Mixed (C)	100	4789	1	100	4951	1	100	3084*	1
F_{10}	1	-	2500	-	0	2500	-	0	2500	-	0
F_{10}	10	Discrete	2500	545633	0.35	2500	545633	0.35	2500	454855	0.35
F_{10}	10	Real	2500	-	0	2500	-	0	2500	-	0
F_{10}	10	Mixed (D)	2500	534428	0.4	2500	534428	0.4	2500	387765	0.5
F_{10}	10	Mixed (C)	2500	527754	0.6	2500	527754	0.6	**2000**	**232445**	**0.95**

Our results extend to constrained MINLP problems. Comparison with MIES on selected benchmark problems shows that the clustering-based EA approach can be competitive, and often outperforms MIES in terms of evaluations needed. Different types of constraint-handling methods show a strong impact on optimization efficiency. In the benchmarks tested the dynamic penalty method generated best results in most of the cases, while the constraint domination and global competitive ranking method performed similarly to each other.

8. REFERENCES

[1] P. Bosman, J. Grahl, and D. Thierens. Benchmarking Parameter-Free AMaLGaM on Functions With and Without Noise. *Evolutionary Computation*, 21(3):445–469, Sept 2013.

[2] P. A. N. Bosman. The Anticipated Mean Shift and Cluster Registration in Mixture-based EDAs for Multi-Objective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO-2010, pages 351–358, New York, NY, USA, 2010. ACM.

[3] P. A. N. Bosman, J. Grahl, and D. Thierens. Enhancing the Performance of Maximum-Likelihood Gaussian EDAs Using Anticipated Mean Shift. In *Parallel Problem Solving from Nature – PPSN X*, LNCS, pages 133–143, 2008.

[4] M. Bussieck and A. Pruessner. Mixed-integer Nonlinear Programming. *SIAG/OPT Viewsand-News*, 14(1):19–22, 2003.

[5] M. T. Emmerich, R. Li, A. Zhang, I. Flesch, and P. Lucas. Mixed-Integer Bayesian Optimization Utilizing A-Priori Knowledge on Parameter Dependences. In *Proceedings of the 20th Belgium–Netherlands Conference on Artificial Intelligence (BNAIC)*, pages 65–72, 2008.

[6] R. Li, M. T. M. Emmerich, J. Eggermont, T. Bäck, M. Schütz, J. Dijkstra, and J. H. C. Reiber. Mixed Integer Evolution Strategies for Parameter Optimization. *Evolutionary Computation*, 21(1):29–64, 2013.

[7] T. Runarsson and X. Yao. Constrained evolutionary optimization. In *Evolutionary Optimization*, volume 48 of *International Series in Operations Research and Management Science*, pages 87–113. Springer US, 2002.

[8] K. L. Sadowski, D. Thierens, and P. A. N. Bosman. Combining Model-Based EAs for Mixed-Integer Problems. In *Parallel Problem Solving from Nature — PPSN XIII*, LNCS, pages 342–351. Springer, 2014.

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

[10] 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.