# Learning and Exploiting Mixed Variable Dependencies with a Model-Based EA

Krzysztof L. Sadowski
Utrecht University
Dept. of Computer Science
Utrecht
The Netherlands
k.l.sadowski@uu.nl

Peter A.N. Bosman
Centrum Wiskunde & Informatica
(CWI)
Amsterdam
The Netherlands
peter.bosman@cwi.nl

Dirk Thierens
Utrecht University
Dept. of Computer Science
Utrecht
The Netherlands
d.thierens@uu.nl

***Abstract*—Mixed-integer optimization considers problems with both discrete and continuous variables. The ability to learn and process problem structure can be of paramount importance for optimization, particularly when faced with black-box optimization (BBO) problems, where no structural knowledge is known a priori. For such cases, model-based Evolutionary Algorithms (EAs) have been very successful in the fields of discrete and continuous optimization. In this paper, we present a model-based EA which integrates techniques from the discrete and continuous domains in order to tackle mixed-integer problems. We furthermore introduce the novel mechanisms to learn and exploit mixed-variable dependencies. Previous approaches only learned dependencies explicitly in either the discrete or the continuous domain. The potential usefulness of addressing mixed dependencies directly is assessed by empirically analyzing algorithm performance on a selection of mixed-integer problems with different types of variable interactions. We find substantially improved, scalable performance on problems that exhibit mixed dependencies.**

***Index Terms*—Mixed-Integer Optimization, Model-building, Evolutionary Algorithms.**

## I. Introduction

The ability to learn and process problem structure is a common challenge in optimization. This ability is especially important for optimizing problems where little information is known a priori about the problem structure. Such problems are known as black-box optimization (BBO) problems and often emerge in real-world applications. BBO approaches found in literature are predominantly concerned with optimization in either the discrete or the continuous domain. However, many real-world problems contain variables from both discrete and continuous domains simultaneously. Such problems are known as mixed-integer (MI) problems, and are often particularly difficult to solve.

Model-based Evolutionary Algorithms (EAs) are a family of optimization algorithms known for their robustness and effectiveness in solving BBO problems. One common approach adopted in EAs to learning the problem structure in a black-box setting is estimating a statistical model which captures interactions between problem variables. In mixed-integer problems dependencies between the discrete and the continuous variables are now also possible. Such inter-domain variable dependencies are one of the factors making mixed-integer problems potentially very difficult to solve. This paper focuses on model-based approaches which enable direct learning, and subsequent exploitation of such dependencies.

A recently introduced Genetic Algorithm for Model-Based mixed-Integer opTimization (GAMBIT) [1] integrates the model-building and sampling ability of a discrete model-based EA with a continuous model-based EA. Because of a well-balanced sampling and a model-updating mechanism, GAMBIT has shown good promise in solving a range of mixed-integer problems. The model-building capabilities of GAMBIT can however be considered to be limited, as mixed inter-dependencies are not directly considered or exploited.

Specifically, GAMBIT represents the problem structure by learning and estimating variable subsets which represent groups of problem variables believed to be dependent on each other, known as the Family-of-Subsets (FOS) denoted by $\mathcal{F} = \{\boldsymbol{F}^0, \boldsymbol{F}^1, \ldots, \boldsymbol{F}^{|\mathcal{F}|-1}\}$. The subsets, denoted $\boldsymbol{F}^i$, can only represent discrete or continuous variables exclusively, as different discrete and continuous optimization mechanisms are used respectively to process them.

In this work, we will extend the FOS structure used with GAMBIT to allow for subsets containing both discrete and continuous variables simultaneously, representing mixed inter-domain dependencies. We consider different approaches to this end for learning, and introduce a novel mechanism to allow GAMBIT to process and exploit the mixed subsets directly.

A comparison of performance with alternative algorithms is also provided. Specifically, we consider another population-based approach: the Mixed Integer Evolution Strategy (MIES) [2] as well as the well-known Solving Constraint Integer Programs (SCIP) algorithm [3]. While other alternatives exist, notably the Non-linear Optimization with Mesh Adaptive Direct Search Algorithm

978-1-5090-0623-6/16/$31.00 

(NOMAD) [4], MIDACO (Mixed Integer Distributed Ant Colony Optimization) [5], GAMS suite of commercial algorithms [6] and others, additional performance comparison is outside of the scope of this paper.

The goal of this paper is to examine the ability of the new mechanism to capture, represent and exploit mixed-variable dependencies and to measure the extent to which it improves the optimization capabilities of GAMBIT. For this, we consider a range of unconstrained problems with different types of dependencies, and compare algorithm performances in terms of number of function evaluations needed. Additionally, we consider a set of constrained problems, and compare our results with the MIES and SCIP algorithms. In order to maximize applicability to the real world, in our implementation we adapt a parameter-free scheme which allows an out-of-the-box execution of GAMBIT.

## II. Background

GAMBIT is based on an integration of two model-based EAs: LTGA and iAMaLGaM, for the discrete and continuous domain respectively. GAMBIT utilizes the model-building abilities of these algorithms with some extensions. Details of the individual algorithms are presented first.

### A. LTGA

The Linkage Tree Genetic Algorithm (LTGA) [7] learns the problem structure by first computing a dependency measure between problem variables. Specifically mutual information (MI) is computed between all pairs of variables at each generation. These values are then used as a distance measure in a hierarchical clustering algorithm. LTGA thereby builds a tree of variable subsets, starting from the leafs in which each variable is a singleton subset. New subsets are created from other ones through merging the nearest subsets, until one subset remains containing all the problem variables. All the subsets created in this process are then considered members of the FOS, and are used for new solution generation.

To generate new solutions, LTGA iterates over all solutions in the population in an attempt to improve them: for each solution the FOS is traversed and each subset is used as a crossover mask between a randomly selected donor from the population and the parent solution. In other words, the values of variables clustered together at a given node of the linkage tree are copied from a donor onto the parent solution. The crossover result is immediately evaluated. If the resulting offspring solution is better or equal than its parent, it instantly replaces the parent. Otherwise, the offspring is discarded. This process is repeated until all the linkage tree nodes are processed.

### B. iAMaLGaM

The Incremental Adapted Maximum-Likelihood Gaussian Model Iterated Density Estimation Evolutionary Algorithm (iAMaLGaM) [8] follows the general estimation-of-distribution algorithm (EDA) paradigm. Specifically, each generation iAMaLGaM estimates a probability distribution from the selected solutions and generates new solutions by sampling the estimated distribution. The probability distribution used in iAMaLGaM is the Gaussian distribution. The mean vector and covariance matrix are estimated incrementally using memory decay on maximum-likelihood estimates. Risk of premature convergence is counteracted by a mechanism which scales up the co-variance matrix when needed. Finally, a so-called Anticipated Mean Shift procedure is implemented to improve behavior in slope-like regions of the search space.

## III. GAMBIT

The Genetic Algorithm for Model-based mixed-Integer opTimization (GAMBIT) is a recently introduced EA designed for the mixed-integer domain [1]. Key features of GAMBIT are (i) clustering algorithm, which allows for better exploration of the search space, (ii) an integrated models mechanism which extends the FOS structure of LTGA to allow discrete as well as continuous subsets, and (iii) balanced interleaved model sampling to create new solutions.

**GAMBIT Overview**

$\mathcal{P} \leftarrow$ GenerateAndEvalRandomPopulation($n$)
**while** $\neg$TerminationCriterionSatisfied **do**
  $\mathcal{C} \leftarrow$ DetermineClusterCenters($\mathcal{P}, k$)
  **for** $j \in \{0, 1, \ldots, k\}$ **do**
    $\mathcal{P}_k \leftarrow$ GrowSubPopFromCenters($\mathcal{P}, \mathcal{C}_k$)
    $\mathcal{O}'_k \leftarrow$ IntegratedModelsMechanism($\mathcal{P}_k$)
  $\mathcal{P} \leftarrow \mathcal{O}'_0 \cup \mathcal{O}'_1 ... \cup \mathcal{O}'_k$

Fig. 1. Pseudo-code overview of GAMBIT. $\mathcal{P}$ represents the population of size $n$. $\mathcal{C}$ is a set of solutions representing $k$ cluster (also called sub-population) centers. $\mathcal{P}_i$ represents the $i$-th sub-population and $\mathcal{O}'_i$ is the offspring generated from sub-population $i$.

### A. Clustering Mechanism

Clustering can be a powerful means of identifying and exploiting different regions of the search space. At the beginning of every generation, the population is clustered into $k$ equally-sized clusters of solutions. Each of the clusters correspond to $k$ instances of the integrated models mechanism, which are responsible for generating offspring solutions. The clustering itself is based on a distance metric which considers both the discrete and continuous variables. Specifically, the distance between solutions is measured using the Euclidean distance. To calculate this distance, binary variables are treated as

real valued, taking on values of 1 or 0. All continuous variables are normalized to a [0;1] range.

To generate clusters, $k$ far apart cluster centers are selected by starting with the current best solution in the population as one cluster center, and then choosing the remaining $k-1$ centers as the furthest away solutions from the already selected cluster centers. With all the centers selected, $k$ sub-populations are formed by assigning to each cluster center the $n/k$ solutions nearest it. In every generation each of these sub-populations is used in a independent instance of the integrated models mechanism, which generates offspring solutions from each cluster. Once all offspring solutions are generated, they form a new population, and the clustering process beings anew.

### B. Integrated Models Mechanism

In order to deal with mixed-integer problems which may be composed of different proportions of discrete and continuous variables, and consist of different inter- or intra- domain variable dependencies, a carefully balanced model-learning and sampling mechanism is needed.

In GAMBIT, this is achieved by expending the FOS structure used with LTGA. Like in LTGA, a linkage tree consisting of discrete variable subsets is constructed, and each of the linkage tree nodes is considered a subset of the FOS. Additionally, continuous subsets are now also present in the FOS. Specifically, $2l_c-1$ continuous subsets are added to the FOS, to maintain a proportionally equivalent number of discrete and continuous subsets. Unlike the discrete subsets, all continuous subsets are the same and contain all continuous problem variables.

The (sub)population to optimize is provided through the clustering mechanism. Each solution consists of a discrete and a continuous component. The core of the algorithm has two nested loops, which iterate over each linkage tree subset, and additionally iterate over every solution for each of the subsets. The parameters of the Gaussian model for the continuous variables is learned after a given discrete subset was applied to each solution. The FOS model, however, is learned only after all the subsets have been tried on all solutions. This process continues until a termination condition is reached. Solution acceptance criteria also differ. The continuous model is learned from the top $\tau$ = 0.35 fraction of the population, following the procedure of iAMaLGaM. When continuous variables are sampled, they are always accepted without any other restrictions. The discrete model is built from the entire population. To generate selection pressure, when a crossover mask is applied, the resulting solution is only accepted if it improves, or has equal fitness following the procedure of LTGA.

### C. Parameter Freedom

The need to specify key parameters prior to execution is a common characteristic of many algorithms. One crucial parameter of many EAs is the population size. While an EA may perform very well for a certain population size on a given problem, setting the population size too small may result in premature convergence and failure to solve the problem, while setting the population size too large may result in very large overhead and inefficiency. When clustering is involved, specifying the number of clusters is often required, and has a significant impact on the performance. GAMBIT needs the specification of both these key parameters: population size and number of clusters. Many algorithms report results only on the optimal parameter settings, which are empirically determined over many runs and with the use of techniques such as bisection search. While applying the optimal parameter settings can be very useful in measuring the algorithms' ability to solve a particular problem, it may often be infeasible to determine the optimal configuration in practice, i.e., considering the black-box setting, and possibly real-world applications with very large and expensive to evaluate objective functions. In order to create a useful algorithm for real-world applications, the ability to run "out of the box", without the need to set parameter values, is very desirable.

A population-size free scheme [9] is used in GAMBIT that removes the need to specify the population size, and extends it to also include the cluster size parameter. In the adaptation of this scheme, an instance of GAMBIT is used as a basis with an initial population size that is determined using iAMaLGaM's minimum population size guideline, adjusted for the presence of discrete and continuous variables: $p_{initialSize} = \sqrt{l_d + l_c}$ and the initial cluster size $k = 1$. When 4 generations of this instance are completed, the current largest population size instance is doubled, and the cluster size is increased by one. One generation of this instance is completed for every 4 generations of the smaller population size. This process of introducing a new instance with a doubled population size, and increased cluster size, continues to happen every 4 generations of the currently biggest instance. If at any generation, the average fitness of a latter instance is better than the average fitness of an earlier population instance, all smaller-sized populations are terminated.

## IV. Inter-Domain Dependency Learning

The presence of mixed-variable dependencies is one of the key difficulties in the mixed domain. The ability to directly learn and address such dependencies promises to be beneficial to optimization.

Previously, GAMBIT used FOS subsets of the same variable type only. As a result, mixed inter-domain dependencies were only addressed indirectly through interleaved model sampling and clustering in the mixed

solution space. In this paper, we introduce the ability to directly deal with mixed-dependencies by allowing FOS subsets that contain both discrete and continuous variables. Two key aspects need to be considered. First is the ability to quantify some measure of strength of dependency between variables of different types, so that subsets representing mixed-variable dependencies can be learned. Second, once we encounter such subset during variation, a mechanism is needed to process and sample new values for all variables in this subset jointly. In this section we focus on the first part. In the next section, we consider the second part.

### A. Mutual Information in the Mixed Space

MI between two random variables is a well known measure of mutual dependence in information theory. LTGA already calculates MI between pairs of discrete variables. We consider three methods of extending the use of MI to solution spaces that contain both discrete and continuous variables.

In our first method, the continuous problem variables are discretized. This is accomplished with a crude discretization that calculates the average value of a variable in the current population, and assigns a zero value for all values below, and a one value for all values above the average. With these temporarily discretized variables, MI can be calculated in the same way as with LTGA, but now for all variables.

In our second method, we do the converse. It is again possible to calculate MI between all variables if the discrete variables are treated as real-valued and using the definition of MI assuming a parametric distribution over the variables such as the normal distribution. For the purposes of this calculation, the continuous variables are normalized to a [0;1] range, and binary variables are treated as real values.

In our third method, we estimate MI without the need to convert one type of variable into another. This is not straightforward, as MI calculated in the discrete and in the continuous domains are not directly comparable. For this, we utilize a recently introduced approach called the nearest neighbor method [10]. This method relies on calculating the Jensen-Shannon divergence, which measures the dissimilarity between two or more continuous probability distributions [11].

The nearest neighbor method aims to determine if different values of the discrete variable $X$ are biased towards different values of the continuous variable $Y$. More specifically, for each data point $(x_i, y_i)$ a number $I_i$ is computed based on the nearest neighbors of $(x_i, y_i)$ considering only the continuous variable $y_i$. First, the $k$-th closest neighbor of point $(x_i, y_i)$ is found whose value of the discrete variable is $x_i$, i.e. point $(x_i, y_j)$ for which $|y_i - y_j|$ is the $k$-th smallest. Using $d$ as the distance to the $k$-th neighbor, the number of neighboring points $m_i$ that are within distance $d$, is counted from the entire data set, where distances again only consider real-valued variables, but now the discrete value is not restricted to match $x_i$. We compute

$$I_i = \psi(N) - \psi(N_{xi}) + \psi(k) - \psi(m_i) \quad (1)$$

where $\psi$ is the digamma function, $N$ is the number of all data points, $N_{x_i}$ is the number of data points whose discrete values is equal to $x_i$. The estimation of the MI between the discrete continuous pair (X,Y) is an average $I_i$ over all the data points:

$$I(X,Y) = \langle I_i \rangle = \psi(N) - \langle \psi(N_x) \rangle + \psi(k) - \langle \psi(m) \rangle \quad (2)$$

In our implementation we use $k = 3$, following a recommended guideline for this method [10].

### B. FOS Subset Selection

We now have the ability to estimate MI for mixed pairs of 2 variables. However, a way to capture variable dependencies for multiple variables in FOS subsets is still needed. A FOS structure can be efficiently constructed with hierarchical clustering, e.g., like in LTGA. This can be expanded to allow for the generation of mixed subsets within the FOS. We consider two intuitive ways of doing so.

Our first approach is a direct extension: we use the same method to create a linkage tree with hierarchical clustering, but consider all problem variables. Although we now have a way to compute MI for pairs of discrete, continuous, or mixed discrete-continuous variables, the notion and range of MI for these different types of pairs is too difficult to build proper linkage trees directly from all these different notions. Depending on various issues, such as continuous correlations strength, such a tree could unfairly focus on one type of variable domain and disregard all other dependencies until the end. For this reason, we consider only using converted variables to compute MI in only one domain, i.e, as discussed in the previous section. A tree obtained in this fashion contains $2(l_c + l_d) - 1$ subsets, and can include mixed subsets. Depending if the MI was estimated entirely in the continuous or discrete domain, we refer to this approach as a fully continuous based FOS (G-Full Cont.) or fully discrete based FOS (G-Full Disc.) respectively.

Our second approach retains existing mechanisms for generating strictly discrete and strictly continuous subsets, and appends it with mixed subsets, which are built using an additional linkage tree. This approach allows us to retain all specific intra-domain dependencies, and join them with specifically learned inter-domain dependencies. In order to generate the strictly mixed-subset linkage tree, the MI between variables of the same type is set to $-\infty$, ensuring that in such a tree, all non-leaf nodes are guaranteed to contain both types of variables in each subset. These non-leaf nodes are appended to the FOS used with the original GAMBIT. This results in

$3l_d+3l_c-3$ subsets: $2l_d+2l_d-2$ from the original GAMBIT subset structure, joined with $l_d + l_c - 1$ non-leaf subsets from the mixed variable linkage tree. Based on this FOS representation, we introduce three additional GAMBIT variants based on how MI is estimated: GAMBIT with the nearest neighbor method (G-Joined NN), GAMBIT with the appended mixed subsets based on continuous estimation (G-Joined Cont.) and discrete estimation (G-Joined Disc.).

## V. Mixed Subset Processing

With the FOS now capable of representing mixed subsets, the ability to process such subsets needs to be added to the integrated models mechanism.

When a mixed FOS subset is encountered, sub-models are created. Specifically, the sub-population that is currently handled by the integrated models mechanism, is grouped into clusters, this time on the sub-population level, using the same mixed distance metric explained in Section IIIA, however now only considering the variables present in the mixed subset for distance calculation. This results in $k$ equally-sized sub-clusters for every mixed subset in the FOS. The value for $k$ is set equal to the number of clusters used by the main clustering mechanism of GAMBIT. This way $k$ does not need to be explicitly specified. A sub-model represents only the variables present in those subsets, making it relatively inexpensive in terms of additional computing resources needed (i.e., computing covariance matrices and decomposing them for sampling).

When a mixed subset is encountered, for each solution a donor solution is randomly selected from the population. The sub-cluster to which this donor belongs, is probabilistically calculated (i.e. maximum likelihood match). Generating a new solution for a mixed subsets samples the discrete and the continuous variables specified by the mixed subset. The discrete subset variables are sampled from the donor, into the current solution. The continuous variables are sampled from the sub-model which was matched to the mixed subset. Only after both, the continuous and discrete variables of the mixed subset have been sampled, changes are evaluated and kept only if it results in an improvement. Otherwise, the change is rejected. This procedure is illustrated in Figure 2.

## VI. Problems

We consider a set of unconstrained and constrained mixed-integer functions designed to exert different types of landscape features.

A Mixed-Integer Problem is 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

**Integrated Model Sampling for a cluster sub-population $\mathcal{P}_k$**

```
INTEGRATEDMODELMECHANISM(P_k)
  F = LearnFOS(P_k)
  for i ∈ {0, 1, ..., (|F| − 1)} do
    if F^i is all continuous then
      S ← TRUNCATIONSELECTION(P_k, τ)
      UPDATEGAUSSIANPARAMETERS(k, S)
      for j ∈ {0, 1, ..., n − 1} do
        ((P_k)_j)_{F^i} ← SAMPLECONTINUOUSMODEL(k, F^i)
        EVALUATEFITNESS((P_k)_j)
    if F^i is all discrete then
      for j ∈ {0, 1, ..., n − 1} do
        O ← (P_k)_j
        donor ← GETRANDOMSOLUTION(P_k)
        (O)_{F^i} ← (donor)_{F^i}
        EVALUATEFITNESS(O)
        if fitness(O) at least as good as fitness((P_k)_j) then
          ((P_k)_j)_{F^i} ← (O)_{F^i}
    if F^i is mixed then
      P_{k_sub}[0, ..., k − 1] ← GROUPINTOCLUSTERS(i, k, P_k)
      UPDATEMIXEDSUBMODELS(i, P_{k_sub})
      for j ∈ {0, 1, ..., n − 1} do
        SubID ← DETERMINESUBPOPULATION(j)
        O ← (P_k)_j
        (O)_{F^i} ← SAMPLESUBMODEL(i, P_{k_sub}[SubID])
        EVALUATEFITNESS(O)
        if fitness(O) at least as good as fitness((P_k)_j) then
          ((P_k)_j)_{F^i} ← (O_k)_{F^i}
  return P_k
```

Fig. 2. Pseudo-code for generating solutions for mixed integer problems with GAMBITs Learning Models.

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

where $d_i \in \{0, 1\}, c_i \in \mathbb{R}$ and $\boldsymbol{x_d}$, $\boldsymbol{x_c}$ are the groups of all discrete and continuous variables, respectively. $f$ is the objective function. **h** and **g** are the sets of equality and inequality constraint functions respectively. If both sets are empty, the mixed-integer problem is said to be unconstrained.

### A. Unconstrained Problems

The first mixed problem $F_1$ is the Onemax-Sphere function. It is a concatenation of two problems where all the variables are completely independent. The DT5-R.Ellipse, or $F_2$, is a problem where subsets of discrete variables are dependent on each other by use of well known deceptive trap functions, while the continuous variables are all strongly dependent on each other due to a parameter rotation of 45 degrees. In our problems we consider trap functions of size 5. Its definition in Table I shows that the global optimum occurs when all the bits are 1's. However, $2^m - 1$ deceptive local optima exist when all bits are 0's or all bits are 1's in a subset. No dependence between the discrete and continuous domains exist in $F_1$ and $F_2$. Function $F_3$ includes intra-domain, as well as inter-domain dependencies. It is a specific combination of the previously defined $F_{DT5}$ function with the rotated ellipsoid. It is additively de-

composable and consists of sub-functions 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 k-variable 45 degree rotated ellipse function corresponds with each of those combinations. The continuous function that 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, function $F_4$ represents only mixed-variable dependencies. Similarly to $F_3$ it is a concatenation of 5 variable blocks, however because of the use of onemax and sphere functions no intra-variable dependencies exist. In this function the Onemax $D^{block}$ correspondents to 32 differently translated sphere functions. This means that the onemax function and the correct sphere function needs to be solved for each block to find the global optimum.

TABLE I
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<br>$F_{Ellip.}(\boldsymbol{x_c}) = \sum_{i=0}^{l_c-1} 10^{6*i/(l_c-1)} * c_i^2$ |
| $F_{Onemax}(\boldsymbol{x_d}) = \sum_{i=0}^{l_d-1} d_i$ |
| $F_{DT5}(\boldsymbol{x_d}) = \sum_{i=0}^{l_d/k-1} f^{sub}_{Trap\ k}(\sum_{j=ki}^{ki+k-1} d_j)$ , where<br>$f^{sub}_{Trap\ k}(u) = \begin{cases} 0 & : if\ u = k \\ 1-(k-1-u)/k & : otherwise \end{cases}$ |

TABLE II
UNCONSTRAINED BENCHMARK PROBLEMS

| ID | Definition |
|---|---|
| $F_1$ | $F_1(\boldsymbol{x_d}, \boldsymbol{x_c}) = F_{Onemax}(\boldsymbol{x_d}) + F_{Sphere}(\boldsymbol{x_c})$ |
| $F_2$ | $F_2(\boldsymbol{x_d}, \boldsymbol{x_c}) = F_{DT5}(\boldsymbol{x_d}) + F_{R.Ellip.}(\boldsymbol{x_c})$ |
| $F_3$ | $F_3(\boldsymbol{x_d}, \boldsymbol{x_c}) = \sum_{i=0}^{0.5l/k-1}(1 + 100 f^{sub}_{trap}(\sum_{j=ki}^{ki+k-1} d_j)) * (1 + f^{sub}_{Ellipse}(D_i^{block}, C_i^{block}))$ |
| $F_4$ | $F_4(\boldsymbol{x_d}, \boldsymbol{x_c}) = \sum_{i=0}^{0.5l/k-1}(1 + 100 f_{Onemax}(D_i^{block}) * (1 + f^{sub}_{Sphere}(D_i^{block}, C_i^{block}))$ |

### B. Constrained Problems

Many industrial mixed-integer problems include constraints. Often, it is the constraints, rather than the objective function that contribute more to the difficulty of a given problem. We therefore also consider a set of mixed-integer problems from the MINLP Benchmark Library [12]. The specification of the objective function, constraints and parameters ranges can be found in Table III. This benchmark set contains the same problems used by [2] for testing MIES. Minimization is assumed for both unconstrained and constrained problems. In GAMBIT, the objective function is adjusted using a standard dynamic penalty method, which allows the exploration of the infeasible space early on, but by increasing the penalty factor over time, gradually puts more focus on the feasible space. Specifically, the penalty factor (a squared sum of all constraint violations) is multiplied by the current generation count causing this dynamic increase in the penalty values for infeasible solutions over time. This default approach is used with all the constrained problems tested, without any adjustments.

## VII. RESULTS

All problems are tested with GAMBIT in the parameterless setting. The success criterion is finding a solution within $10^{-10}$ of the known global optimum at least 29 out of 30 times. A run is considered unsuccessful if the optimum is not found after 10 million evaluations or execution time exceeds an hour. A successful run for any tested algorithm was usually completed within a few minutes. The reported results for the unconstrained problems consist of the same number of discrete and continuous variables ($l_c = l_d = l/2$), while the constrained problems vary in variable composition as specified in Table III.

Figure 3 shows that GAMBIT does not benefit from any variant of the mixed-dependency processing mechanism on the $F_2$ problem, where no inter-variable dependencies exist, however it does on the $F_1$ problem. This can be attributed to the larger number of subsets that exist per generation when the mixed dependency learning mechanisms are used. Because the subsets are correct, more subsets means more selection pressure used to get better solutions faster. Since all problem variables are independent, modeling dependencies over generations is not necessary.

On problems that contain inter-domain dependencies ($F_3$ and $F_4$) the advantages of the newly introduced mechanism become apparent. All variants of GAMBIT with the mixed subset processing mechanisms outperform the original GAMBIT. Approaches that use discrete MI estimation (G-Full Disc. and G-Joined Disc.) perform worse than the remaining alternatives. This is likely due to the crude one bit discretization used in these approaches. The G-Joined NN is the most efficient variant. This suggests that calculating MI in exclusively discrete or continuous domain results in a less accurate estimation of mixed dependencies than with the nearest-neighbor approach.

Our observations are further confirmed on the constrained problems (see Table V). Dependency processing is less useful on problems with few variables. This can

TABLE III
SPECIFICATIONS OF THE MINLP SELECTED CONSTRAINED BENCHMARK PROBLEMS

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

be seen in functions $F_5$ and $F_9$, where learning the mixed subsets is not advantageous. Furthermore, the G-Full Disc. and G-Full Cont. approaches fail to solve $F_{10}$. The joint tree approaches are successful however, and improve on the original "NoMixedLearning" variant on most problems. This suggests that relying on a single tree based FOS may be insufficient as intra-domain dependency information may be lost. However, when the mixed dependency information is considered separately, and joined with the individual domain dependency knowledge, the results show an increase in performance. The G-Joined NN variant outperforms all alternatives on $F_6$, $F_7$, $F_9$ and $F_{10}$, and will be used with GAMBIT from now on.

TABLE IV
COMPARISON OF MIES, SCIP AND GAMBIT ON PROBLEMS $F_1$-$F_{10}$. ABILITY TO SOLVE ALL TESTED PROBLEM SIZES IS REPRESENTED BY ✓

| Alg. | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **MIES** | ✓ | x | x | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **SCIP** | ✓ | ✓ | x | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **GAMBIT** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Additionally, we consider the performance of two known mixed-integer solvers: MIES and SCIP. MIES is a mixed-integer extension to the well known $(\mu + \lambda)$-ES for continuous problems to mixed-integer search spaces [2]. The Solving Constraint Integer Programs (SCIP) algorithm [3] is an efficient branch-and-bound MINLP solver. All three algorithms succeed in solving the constrained problems. Unconstrained problems were tested with all three algorithms on problem sizes of length 20, 40, 80 and 160. The results are summarized in Table IV. Only GAMBIT is capable of consistently solving all functions presented in this paper. MIES does not solve $F_2 - F_4$, while SCIP fails to solve $F_3$ and $F_4$. These results suggest that dealing with landscape features created by trap functions and rotated ellipsoids ($F_2$) or inter-domain variable dependencies ($F_3$, $F_4$) are difficult for these algorithms to deal with, which further highlights the usefulness of model-based approaches for mixed-integer problems that contain such features. It should further be noted that although SCIP solves many of the problems, it relies heavily on a mathematical input format that can exploited efficiently. Black-box simulation could thus not be tackled (not a restriction for MIES and GAMBIT).

## VIII. CONCLUSIONS

The potential existence of mixed-variable dependencies is one of the main factors why mixed-integer problems are considered challenging. In this paper we studied the potential benefits of adding a mixed inter-variable dependency learning and processing mechanism to a previously introduced Genetic Algorithm for Model-Based mixed-Integer opTimization (GAMBIT). For the dependency learning we considered three mutual information (MI) based approaches to estimate a measure of dependency between all pairs of variables. We estimated the mixed variable dependencies using an entirely discrete, entirely continuous and a mixed nearest-neighbor based statistical approach. We introduced an extended Family-of-Subsets (FOS) structure, which allowed us to represent not only exclusively discrete and continuous variable subsets, but also mixed ones.

We showed that while a mechanism for explicitly learning and exploiting mixed dependencies can generate some unnecessary overhead on problems where no mixed-dependencies are present, it improves performance on more difficult problems containing such dependencies. On our benchmark set, the most successful variant of the algorithm included the nearest-neighbor MI estimation method, which on the majority of problems outperformed our alternative approaches.

Real-world applicability is very important. With a parameter-free scheme and the inclusion of our novel dependence learning and processing mechanism, we showed that GAMBIT is capable of solving a range of unconstrained and constraint problems in a black-box

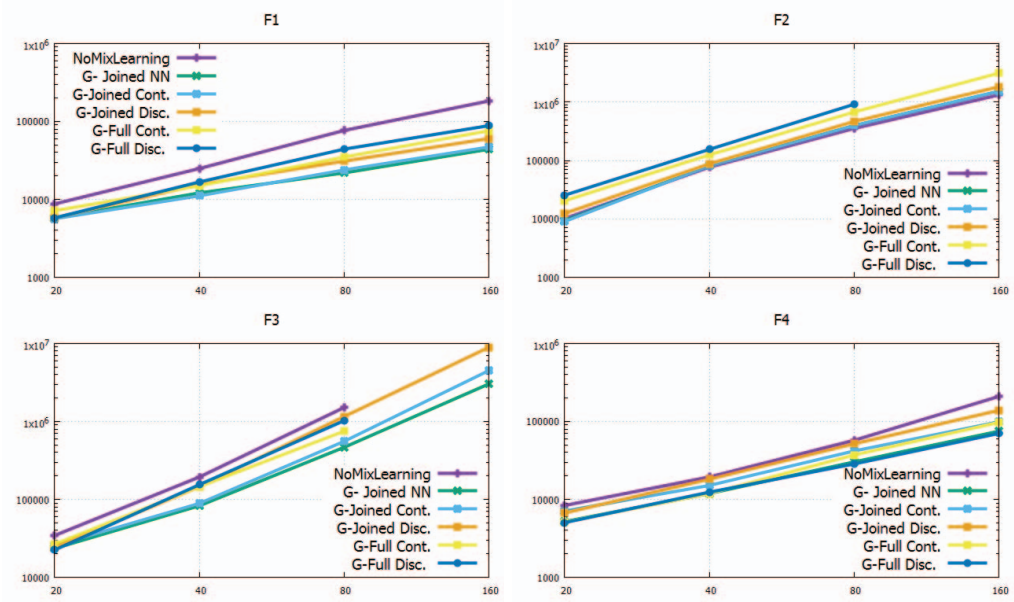


Fig. 3. Average number of evaluations required by GAMBIT variants on unconstrained benchmarks $F_1$-$F_4$, over 30 runs

TABLE V
PERFORMANCE OF GAMBIT VARIANTS ON CONSTRAINT PROBLEMS (AVERAGE NUMBER OF EVALUATIONS REQUIRED, OVER 30 RUNS)

| | GAMBIT Variant | | | | | |
|---|---|---|---|---|---|---|
| **Problem** | **No Mixed Learning** | **G-Joined NN** | **G-Joined Cont.** | **G-Joined Disc.** | **G-Full Cont.** | **G-Full Disc.** |
| **F5** | **1056** | 4096.7 | 4521 | 5836.3 | 4939 | 4015.8 |
| **F6** | 50431.5 | **30868.3** | 35976 | 35724.6 | 41953.2 | 62347.2 |
| **F7** | 3404.8 | 2909 | 2911 | 2819.1 | 3388.4 | **2093.9** |
| **F8** | 114887.2 | **54645.3** | 57855 | 75241.6 | 58098.1 | 119347.4 |
| **F9** | 6271.6 | **4641.7** | **4693.3** | 6709.5 | **4771.6** | **4892.7** |
| **F10** | 720494 | **418224** | 558219 | 6311296 | x | x |

setting, without the need to specify any parameters such as population size or number of clusters. The performance of GAMBIT was compared with two other mixed-integer solvers: MIES and SCIP. Both of these solvers succeeded in optimizing the constrained problems, however failed to consistently solve the unconstrained problems with underlying mixed-dependencies. GAMBIT was able to solve all the problems tested. Our results show the potential and capacity of learning and exploiting mixed-variable dependencies of mixed-integer problems with model-based evolutionary approaches, and provide a good foundation and framework for possible future real-world applications.

## REFERENCES


[1] K. L. Sadowski, P. Bosman, and D. Thierens, "A Clustering-Based Model-Building EA for Optimization Problems with Binary and Real-Valued Variables," in *Proceedings of the 2015 Genetic and Evolutionary Computation Conference*, GECCO '15, pp. 911–918, ACM, 2015.

[2] 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*, vol. 21, no. 1, pp. 29–64, 2013.

[3] T. Achterberg, "SCIP: Solving Constraint Integer Programs," *Mathematical Programming Computation*, vol. 1, pp. 1–41, 2009.

[4] S. Le Digabel, "Algorithm 909: NOMAD: Nonlinear Optimization with the MADS algorithm," *ACM Transactions on Mathematical Software*, vol. 37, no. 4, pp. 1–15, 2011.

[5] M. Schlüter, J. A. Egea, and J. R. Banga, "Extended ant colony optimization for non-convex mixed integer nonlinear programming," *Computers and Operations Research*, vol. 36, no. 7, pp. 2217 – 2229, 2009.

[6] M. Bussieck and A. Meeraus, "General Algebraic Modeling System (GAMS)," in *Modeling Languages in Mathematical Optimization*, vol. 88 of *Applied Optimization*, pp. 137–157, Springer US, 2004.

[7] D. Thierens and P. A. N. Bosman, "Optimal Mixing Evolutionary Algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '11, pp. 617–624, ACM, 2011.

[8] P. A. N. Bosman, J. Grahl, and D. Thierens, "Benchmarking Parameter-free AMaLGaM on Functions with and Without Noise," *Evol. Comput.*, vol. 21, pp. 445–469, Sept. 2013.

[9] G. R. Harik and F. G. Lobo, "A Parameter-less Genetic Algorithm," in *IEEE transactions on evolutionary computation*, pp. 523–528, 1999.

[10] B. C. Ross, "Mutual Information between Discrete and Continuous Data Sets," *PLoS ONE*, vol. 9, 02 2014.

[11] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating Mutual Information," *Phys. Rev. E*, vol. 69, Jun 2004.

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