Solving Satisfiability in Fuzzy Logics by Mixing CMA-ES

Tim Brys
Artificial Intelligence Lab, VUB
Pleinlaan 2
1050 Brussels, Belgium
timbrys@vub.ac.be

Madalina M. Drugan
Artificial Intelligence Lab, VUB
Pleinlaan 2
1050 Brussels, Belgium
mdrugan@vub.ac.be

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

Martine De Cock
Dept. of Applied Math., Comp. Sc. and Stat., UGent
Krijgslaan 281 (S9)
9000 Gent, Belgium
martine.decock@ugent.be

Ann Nowé
Artificial Intelligence Lab, VUB
Pleinlaan 2
1050 Brussels, Belgium
anowe@vub.ac.be

ABSTRACT

Satisfiability in propositional logic is well researched and many approaches to checking and solving exist. In infinite-valued or fuzzy logics, however, there have only recently been attempts at developing methods for solving satisfiability. In this paper, we propose new benchmark problems and analyse the function landscape of different problem classes, focussing our analysis on plateaus. Based on this study, we develop Mixing CMA-ES (M-CMA-ES), an extension to CMA-ES that is well suited to solving problems with many large plateaus. We empirically show the relation between certain function landscape properties and M-CMA-ES performance.

Categories and Subject Descriptions

G.1 [**Numerical Analysis**]: Optimization

General Terms

Algorithms, Performance

Keywords

CMA-ES, Fuzzy Satisfiability, Mixing

1. INTRODUCTION

A logical formula, or a set of formulas, is said to be satisfiable if there exists a truth assignment to its variables that makes every formula true. Satisfiability checking is verifying whether such an assignment exists, and satisfiability solving means finding such an assignment. This problem is known as SAT in propositional logic and is of interest to researchers from various domains, as many problems can be reformulated as a SAT problem and subsequently solved by a state-of-the-art SAT solver.

In fuzzy logics – logics with an infinite number of truth degrees – the same principle of satisfiability exists (SAT_∞). Like its classical counterpart, it is useful for solving a variety of problems. Indeed, many fuzzy reasoning tasks can be reduced to SAT_∞, including reasoning about vague concepts in the context of the semantic web [11], fuzzy spatial reasoning [9] and fuzzy answer set programming [7], which in itself is an important framework for non-monotonic reasoning over continuous domains (see e.g. [8, 13]). Importantly, the transition into fuzzy logics makes SAT_∞ a problem on a continuous domain, instead of a discrete (boolean) domain.

Solving satisfiability in fuzzy logics has however received much less attention than its counterpart in classical logics. In [4], Hähnle proposes a mixed integer programming (MIP) approach for Łukasiewicz logic, which unfortunately suffers from scalability issues that are inherent to MIP. Schockaert et al. also propose a solver for SAT_∞ in Łukasiewicz logic [10], which reduces the infinite-valued logic to a finite-valued one and then applies a constraint satisfaction solver to check satisfiability. This approach suffers from an exponential complexity in the granularity of the truth values in a problem. Recently, we proposed a new solver [2] that models satisfiability as an optimization problem over a continuous domain and has the advantage of being independent of the underlying logic and its operators, as well as not suffering from the complexity issues associated with analytical solvers, as our results showed. The disadvantage of this approach is that it cannot be a complete solver, i.e. deciding both SAT_∞ and UNSAT_∞, unless the optimization algorithm is proven to converge to the global optimum. The currently best performing optimization algorithm on these problems is the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), which is considered state-of-the-art in black-box optimization on a continuous domain.

In this paper, we define an objective function for SAT_∞ and intuitively show that optimizing this function equals solving SAT_∞. We incrementally describe the generation of a set of challenging benchmark problems representing different problem classes from different logics and motivate our choices. Using landscape analysis, we identify certain properties from these problem classes and use this knowledge to propose an extension to CMA-ES, aimed to improve its performance. In Sections 2 and 3, we introduce Fuzzy Logics and CMA-ES respectively. Subsequently, in Section 4, we introduce the first problem classes and perform a landscape analysis. In Section 5, we propose an extension to CMA-ES and evaluate it on the benchmark problems, and in Section 6, we introduce two new problem classes and perform the same analysis.

2. FUZZY LOGIC AND SAT$_\infty$

In fuzzy logics [5], truth is expressed as a real number taken from the unit interval $[0, 1]$. Essentially, there are an infinite number of truth degrees possible. A formula in fuzzy logics is built from a set of variables $\mathcal{V}$, constants taken from $[0, 1]$ and n-ary connectives for $n \in \mathbb{N}$. An interpretation is a mapping $\mathcal{I} : \mathcal{V} \to [0, 1]$ that maps every variable to a truth degree. We can extend this fuzzy interpretation $\mathcal{I}$ to formulas as follows:

- For each constant c in $[0, 1]$, $[c]_\mathcal{I} = c$.
- For each variable v in $\mathcal{V}$, $[v]_\mathcal{I} = \mathcal{I}(v)$.
- Each n-ary connective f is interpreted by a function $\texttt{f} : [0, 1]^n \to [0, 1]$. Furthermore we define

  $$[f(\alpha_1, \ldots, \alpha_n)]_\mathcal{I} = \texttt{f}([\alpha_1]_\mathcal{I}, \ldots, [\alpha_n]_\mathcal{I})$$

  for formulas α_i with $1 \le i \le n$.

The connectives in fuzzy logics typically correspond to connectives from classical logic, such as conjunction, disjunction, implication and negation, which are interpreted respectively by a t-norm, a t-conorm, an implicator and a negator. A triangular norm or t-norm $\texttt{T}$ is an increasing, associative and commutative $[0, 1]^2 \to [0, 1]$ mapping that satisfies the boundary condition $\texttt{T}(1, x) = x$ for all x in $[0, 1]$. Similarly, a triangular conorm or t-conorm $\texttt{S}$ is an increasing, associative and commutative $[0, 1]^2 \to [0, 1]$ mapping that satisfies the boundary condition $\texttt{S}(0, x) = x$. An implicator $\texttt{I}$ is a $[0, 1]^2 \to [0, 1]$ mapping that is decreasing in its first argument, increasing in its second argument and that satisfies the properties $\texttt{I}(0, 0) = \texttt{I}(0, 1) = \texttt{I}(1, 1) = 1$ and $\texttt{I}(1, 0) = 0$. A negator $\texttt{N}$ is a decreasing $[0, 1] \to [0, 1]$ mapping that satisfies $\texttt{N}(0) = 1$ and $\texttt{N}(1) = 0$.

As an example of a particularly popular fuzzy logic, in Łukasiewicz logic, negation $\neg$, conjunction $\otimes$, disjunction $\oplus$ and implication $\to$ are interpreted as follows:

- $[\neg\alpha]_\mathcal{I} = 1 - [\alpha]_\mathcal{I}$
- $[\alpha \otimes \beta]_\mathcal{I} = \max([\alpha]_\mathcal{I} + [\beta]_\mathcal{I} - 1, 0)$
- $[\alpha \oplus \beta]_\mathcal{I} = \min(1, [\alpha]_\mathcal{I} + [\beta]_\mathcal{I})$
- $[\alpha \to \beta]_\mathcal{I} = \min(1 - [\alpha]_\mathcal{I} + [\beta]_\mathcal{I}, 1)$

for formulas α and β.

An interpretation $\mathcal{I}$ is said to be a model of a set of formulas Θ iff $\forall \alpha \in \Theta : l \le [\alpha]_\mathcal{I} \le u$, given lower and upper bounds l and u for that formula (usually u is 1, and in classical logic even both l and u are 1). An example of a formula with three variables v_1, v_2, and v_3 in Łukasiewicz logic, with bounds is:

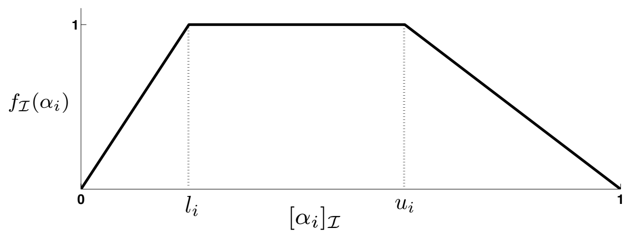


Figure 1: $f_\mathcal{I}(\alpha_i)$ for a formula α_i with lower bound l_i, upper bound u_i and degree of satisfaction $[\alpha_i]_\mathcal{I}$.

$$0.5 \le \neg(v_1 \otimes v_2 \otimes \neg v_3) \le 1 \tag{1}$$

One can easily verify that $\mathcal{I}_1$ with $\mathcal{I}_1(v_1) = 0$, $\mathcal{I}_1(v_2) = 0$ and $\mathcal{I}_1(v_3) = 1$ is a model of this formula as $[\neg(v_1 \otimes v_2 \otimes \neg v_3)]_{\mathcal{I}_1} = 1$. Similarly, $\mathcal{I}_2$ with $\mathcal{I}_2(v_1) = 0.6$, $\mathcal{I}_2(v_2) = 0.7$ and $\mathcal{I}_2(v_3) = 0.2$ is a model too because $[\neg(v_1 \otimes v_2 \otimes \neg v_3)]_{\mathcal{I}_2} = 0.9$. Even though the formula is not perfectly satisfied under interpretation $\mathcal{I}_2$, the degree of satisfaction is still high enough to meet the lower bound $l = 0.5$. The existence of the models $\mathcal{I}_1$ and $\mathcal{I}_2$ show that formula (1) is satisfiable. Solving SAT$_\infty$ amounts to finding a model for the set of formulas given, or deciding that there is no interpretation that satisfies all formulas and that the set is UNSAT$_\infty$.

2.1 SAT$_\infty$ as an optimization problem

As we will investigate an optimization approach to solving satisfiability in fuzzy logics, we need to reformulate SAT$_\infty$ instances as optimization problems, i.e. defining a function over the solution space such that optimizing this function corresponds to solving the SAT$_\infty$ instance. A SAT$_\infty$ problem consists of a set Θ of formulas α_i, each of which must be satisfied to a certain degree, as defined by an upper and lower bound per formula. Given these m formulas α_i, bounds (l_i, u_i), and an interpretation $\mathcal{I}$, we define the objective function f as follows:

$$f(\mathcal{I}) = \frac{\sum_{i=1}^{m} f_\mathcal{I}(\alpha_i, l_i, u_i)}{m} \tag{2}$$

and,

$$f_\mathcal{I}(\alpha_i, l_i, u_i) = \begin{cases} 1 & \text{if } l_i \le [\alpha_i]_\mathcal{I} \le u_i \\ \frac{[\alpha_i]_\mathcal{I}}{l_i} & \text{if } [\alpha_i]_\mathcal{I} < l_i \\ \frac{1-[\alpha_i]_\mathcal{I}}{1-u_i} & \text{if } [\alpha_i]_\mathcal{I} > u_i \end{cases} \tag{3}$$

with $[\alpha_i]_\mathcal{I}$ representing the degree of satisfaction of formula α_i under interpretation $\mathcal{I}$. Each $f_\mathcal{I}$ is a trapezoid function, with a plateau of value 1 when formula α_i's degree of satisfaction lies between the given bounds, and a slope leading to the plateau when the satisfaction lies outside these bounds, as visualised in Figure 1.

This function is formulated such that the global maxima will always have a function value 1 if the SAT$_\infty$ instance is satisfiable. In that case, every global maximum corresponds to a model of the problem.

3. CMA-ES

The Covariance Matrix Adaptation-Evolution Strategy (CMA-ES)[6] is an algorithm belonging to the class of Evolution Strategies, and is considered to be state-of-the-art in black-box optimization on a continuous domain. CMA-ES is a $(\mu/\mu, \lambda)$-ES with the addition of de-randomized adaptation of the multivariate distribution that generates candidate solutions. This additional mechanism allows CMA-ES to efficiently explore promising search directions by adapting the distribution to the local shape of the search space. For a full description of this algorithm, we refer the reader to [6].

In [3], we determined that the restart strategy proposed by Auger and Hansen [1] does not perform well on the problems considered in this paper. Therefore, we use the simple restart strategy that was found to increase performance, i.e. restart after $i \times \lambda$ function evaluations without improvement. It was previously found that $i = 1000$ gives good results.

Furthermore, since we are dealing with box-constraints ($[0, 1]^n$), we need to handle constraint violations. We use a penalty function when the constraint is violated. The penalty is dependent on the distance between the candidate solution and the box. The new objective function is:

$$f(\mathcal{I}) = \begin{cases} \frac{-\sum_{i=1}^{m} f_{\mathcal{I}}(\alpha_i, l_i, u_i)}{m}, & \text{if } \mathcal{I}(\mathcal{V}) \in [0,1]^n \\ 10^4 \sum_{i=1}^{m} \theta(|(\mathcal{I}(v_i) - 0.5)| - 0.5)(\mathcal{I}(v_i) - 0.5)^2, & \\ & \text{if } \mathcal{I}(\mathcal{V}) \notin [0,1]^n \end{cases} \tag{4}$$

with θ the heaviside function (0 for negative input, otherwise 1) and n the number of variables.

We make the new function conditional instead of adding the penalty to the objective function as is commonly done, because numbers beyond [0,1] have no meaning in fuzzy logics and prevent the formulas from being evaluated. Since CMA-ES is a minimization technique, we negate the objective function.

4. PROBLEM CLASSES AND LANDSCAPE ANALYSIS

To evaluate the performance of the proposed algorithms for solving SAT$_\infty$, we need a set of challenging problem instances. In this section we first describe how we built a set of benchmark problems and then perform landscape analysis to identify properties that may give insights into algorithm performance.

4.1 Problem instances

Schockaert et al. [10] propose a procedure to generate satisfiability problems in Łukasiewicz logic; a somewhat different process than what is usual in classical SAT. This is due to the fact that not all formulas in Łukasiewicz logic can be converted to conjunctive-normal form, and that, when restricted to formulas in conjunctive-normal form (in the sense that only lower bounds are used, formulas are composed of conjunctions and negations, and all negations occur immediately in front of atoms), the satisfiability problem in Łukasiewicz logic can be reduced to linear programming, and can thus be decided in polynomial time. Hence, to generate interesting test problems, it is crucial to consider formulas which are not in conjunctive-normal form.

The recursive procedure *form*(p) is defined to generate logical formulas with a fixed number of variable occurrences. The base case, i.e. one variable in a formula, is as follows:

$$\mathit{form}(1) = \begin{cases} v_i & \text{probability } 0.5 \\ \neg v_i & \text{probability } 0.5 \end{cases} \tag{5}$$

where v_i is a randomly chosen variable from a predefined set of variables $\mathcal{V} = \{v_1, v_2, ..., v_m\}$.

The general case, for $p > 1$, is:

$$\mathit{form}(p) = \begin{cases} form(p_1) \otimes form(p_2) & \text{probability } 0.5 \\ \neg(form(p_1) \otimes form(p_2)) & \text{probability } 0.5 \end{cases} \tag{6}$$

with p_1 a random integer between 1 and $p - 1$, and $p_2 = p - p_1$.

The creation of challenging problems depends very much on the selection of bounds for formulas. Schockaert et al. propose the following procedure to select bounds for formula α:

- Let λ_1 be the largest value from $\mathbb{T}_4 = \{0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1\}$ smaller than $[\alpha]_{\mathcal{I}_1}$.
- Let λ_2 be the smallest value from $\mathbb{T}_4$ larger than $[\alpha]_{\mathcal{I}_2}$.
- With probability 0.5, add $\lambda_1 \leq \alpha \leq 1$ to the set of formulas, otherwise add $\lambda_2 \leq \alpha \leq 1$.

The set of benchmark problems used in [10] was generated using this method, with the additional constraint that any variable can occur at most once in every formula. Each problem instance consists of 100 formulas ($n = 100$), with five variable occurrences per formula ($p = 5$), so we can say we are solving 5-SAT$_\infty$. 50 problem instances were generated with 40 variables each.

In [2], we generated an additional set of problem instances, with problem bounds drawn from $\mathbb{T}_{100}$, i.e. $\{0, \frac{1}{100}, \frac{2}{100}, ..., 1\}$ instead of $\mathbb{T}_4$[1]. We distinguish between problems with formula bounds drawn from $\mathbb{T}_4$ and $\mathbb{T}_{100}$, as we have shown that this difference has a strong impact on the performance of current analytical solvers, and might also influence our optimization algorithms. That aside, the use of sets $\mathbb{T}_k$ is purely artificial; in reality, formula bounds are problem dependent and will usually not be directly derived from some set $\mathbb{T}_k$. Determining the smallest k such that $\mathbb{T}_k$ includes all bounds in an instance may not be trivial and will typically yield very large k, which is problematic for the analytical solver from [10]. Still, we use these sets to analyse the impact of granularity on algorithm performance, and because there is no other method known to generate hard SAT$_\infty$ instances.

The two problem classes discussed will from now on be refered to as Ł$_4$ and Ł$_{100}$ respectively.

4.2 Plateaus analysis

A plateau is a collection of points in the search space that span an ℓ-dimensional manifold, with the same fitness for all points. Since most optimization algorithms, including CMA-ES, use gradient information to optimize an objective function, plateaus, which provide zero gradient information,

[1]These benchmark problems, others, and Java code to interpret them can be downloaded here: http://ai.vub.ac.be/members/tim-brys

can hinder performance. We have seen that our objective function is built to yield a plateau for each formula when its degree of satisfaction lies between the bounds specified, and furthermore, the conjunction operator in Łukasiewicz logic calculates the maximum of two values, which also returns a plateau for a range of the input values. Therefore, we can expect our objective function to contain plateaus. We will analyse the two defined problem classes, looking for plateaus.

This analysis is performed by generating random walks. Specifically, for each of the 100 problem instances we have, we perform 10 random walks of 10^6 steps. One step consist of adding ± 0.01 to a randomly picked variable. This value is the smallest granularity in the problems.

Using these random walks, we estimate the size of plateaus by counting the number of subsequent steps of the same fitness. Figure 2(a) plots the relation between plateau size and the number of times they were encountered, which is an estimation of their overall frequency. Both Łukasiewicz problem classes have similar numbers of plateaus, which decrease with plateau size, i.e. there are many more small plateaus than there are large ones.

The relation between plateau size and plateau fitness is plotted in Figure 2(b). Interestingly, fitness increases much with plateau size on $Ł_4$ problems, which means that in general, the larger a plateau, the closer in function value it will be to optimal solutions. We expect this property to ensure that escaping large plateaus needs only be done a few times. This trait is shared with regular SAT landscapes. On the contrary, plateau fitness does not rise much with plateau size on the $Ł_{100}$ problems, which means that escaping from a plateau does not guarantee nearness to optimal solutions, that large plateaus are more distributed with respect to optimal solutions. Note that the plots become more noisy for large plateaus simply because fewer of such plateaus were found.

Given this analysis, we expect that CMA-ES's performance on these problem classes, and specifically on the $Ł_{100}$ problems, may be hindered by these plateaus. Therefore, we propose an additional mechanism for CMA-ES that may help it escape from plateaus.

5. MIXING CMA-ES

The idea we use in this paper for the plateau-escaping mechanism is an application of optimal mixing evolutionary algorithms [12] on the multi-population level: we have multiple CMA-ES populations running in parallel, and we recombine their distributions if this leads to improvements. We call this Mixing CMA-ES or M-CMA-ES. This recombination of distributions is executed by exchanging elements, i.e. variable assignments, between the means of the two populations. Pseudocode describing this mixing can be found in Algorithms 1 and 2.

We start with multiple CMA-ES algorithms running in parallel. When all populations have executed a regular CMA-ES iteration, we pair each one with another and apply mixing to the means of their distributions. The mean is used to evaluate the population's fitness during mixing, as it is the best estimator for the optimum in the population. Mixing goes as follows:

- For each pair of populations, the set of problem variables is randomly divided into s disjunct sets of variables of roughly the same size (the number of variables divided by s is not always a natural number).
- The variables of one of the sets are exchanged between the two population means.
- If at least one of the two new means has greater fitness than both parent means, the exchange is accepted. This ensures global improvement through mixing, avoiding local oscillations when optimized substructures go back and forth between populations.
- This exchange is performed for each disjunct variable set, each time continuing with the accepted means from the previous step.

This is basically optimal mixing with the Marginal Product structure [12], although without structure learning. This is not feasible due to the small number of populations that is practically possible.

Besides the mean of the distribution, CMA-ES also maintains a covariance matrix and evolution paths for the sampling and updating of this distribution. In our mixing, we

Algorithm 1 Mixing CMA-ES

```
 1: procedure M-CMA-ES
 2:     populations ← InitializeCMAESPopulations(n)
 3:     while !stopcriterion do
 4:         for $i \in \{0, 1, .., n\}$ do
 5:             ExecuteCMAESIteration(population$_i$)
 6:         end for
 7:         for $i, j \in \{0, 1, .., n\}$ do
 8:             Mixing(populations$_i$, populations$_j$)
 9:         end for
10:     end while
11: end procedure
```

Algorithm 2 Mixing Two Populations

```
 1: procedure MIXING(MEAN$_1$, MEAN$_2$)
 2:     variableSets ← DisjunctVariableSets(s)
 3:     fp$_1$ ← fitness(mean$_1$)
 4:     fp$_2$ ← fitness(mean$_2$)
 5:     child$_1$ ← mean$_1$
 6:     child$_2$ ← mean$_2$
 7:     for each set ∈ variableSets do
 8:         for each variable ∈ set do
 9:             child$_1$[variable] ← mean$_2$[variable]
10:             child$_2$[variable] ← mean$_1$[variable]
11:         end for
12:         fc$_1$ ← fitness(child$_1$); fc$_2$ ← fitness(child$_2$)
13:         if (fc$_1$ < fp$_1$ and fc$_1$ < fp$_2$) or
                (fc$_2$ < fp$_1$ and fc$_2$ < fp$_2$) then
14:             mean$_1$ ← child$_1$; mean$_2$ ← child$_2$
15:             fp$_1$ ← fc$_1$; fp$_2$ ← fc$_2$
16:         else
17:             for each variable ∈ set do
18:                 child$_1$[variable] ← mean$_1$[variable]
19:                 child$_2$[variable] ← mean$_2$[variable]
20:             end for
21:         end if
22:     end for
23:     RecombineCovAndEvPaths(mean$_1$, mean$_2$)
24: end procedure
```

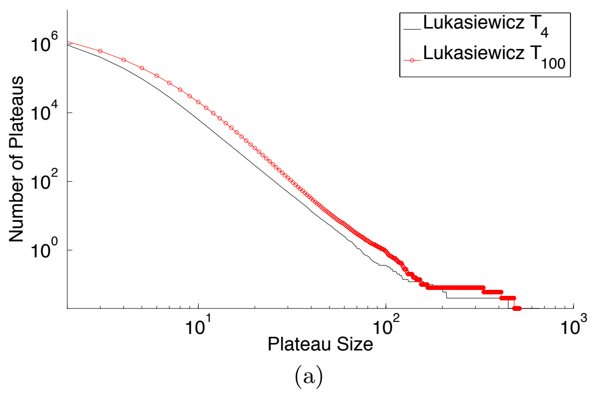
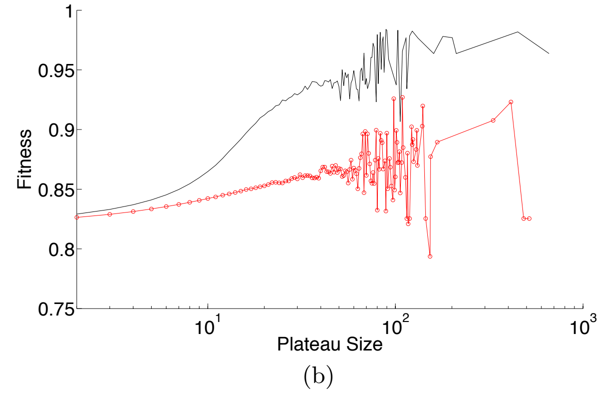
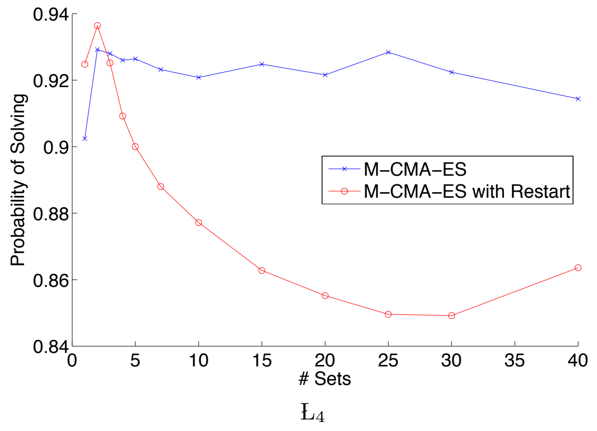


Figure 2: (a) The number of plateaus as a function of plateau size, as estimated by the number of random steps at the same fitness value. (b) The fitness of plateaus as a function of plateau size.





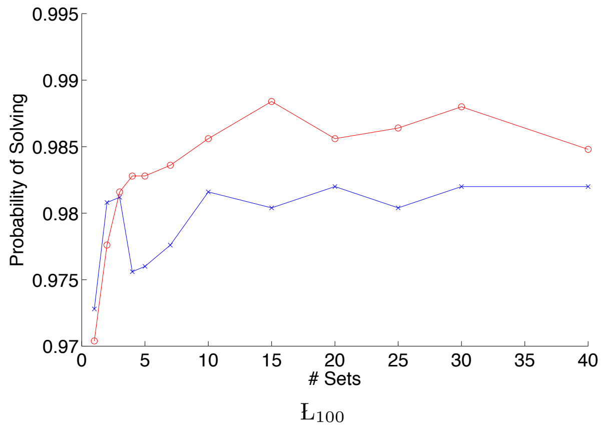
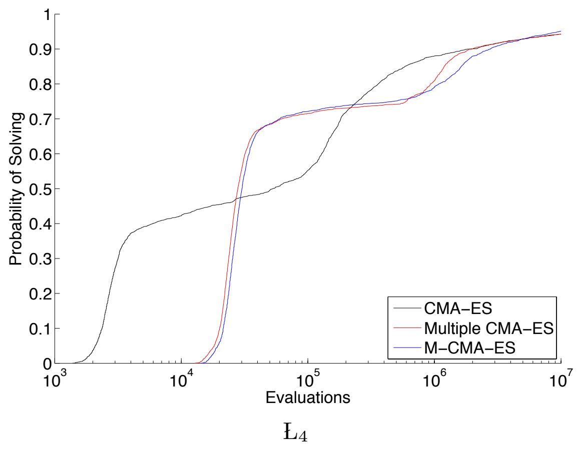


Figure 3: Success ratio achieved given a budget of 10^7 function evaluations as a function of the number of marginal product sets used in mixing on the Łukasiewicz problem classes.



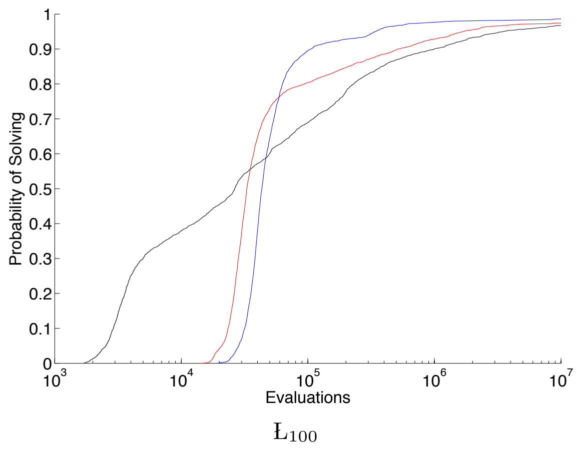


Figure 4: Run length distributions of CMA-ES, multiple CMA-ES and M-CMA-ES on the Łukasiewicz problem classes.

recombine these as described below, under the assumption that we should keep CMA-ES as intact as possible, since these parameters are adapted to the variable values in the mean that are exchanged. Still, we do not test whether this assumption really holds and whether mixing these parameters as well is necessary.

The recombination of the evolution paths is trivial, as these are represented by vectors in which the corresponding values can simply be exchanged. The recombination of the covariance matrix is less simple, as it encodes information between variables that may or may not be kept together during recombination. The way the matrix crossover is implemented is as follows: if variables v_i and v_j stay together, the covariance information for element $(i,\ j)$ comes from the corresponding parent. If the variables come from different solutions, the average of the covariance information from both solutions is calculated and stored. This covariance matrix recombination can only take place after every variable set is exchanged, since only then can we know of all variables whether they stay together or not.

5.1 Empirical validation

Before we empirically evaluate and compare CMA-ES with M-CMA-ES on the given problem classes, we optimize the parameter governing the number of disjunct sets used for mixing. We run two variants of M-CMA-ES , one with and one without the restarting strategy described before. 50 runs of maximum 10^7 evaluations are performed for each instance, resulting in 5000 runs per parameter setting. 10 populations are used, and each population has $\mu = 2$ and $\lambda = 20$, as determined to be near-optimal settings for a single population on these problems [3]. Figure 3 shows the results of this experiment. Note that setting the sets parameter to 1 reduces M-CMA-ES to 10 independent CMA-ES populations (no extra evaluations are performed for mixing). Increasing the number of sets means more evaluations spent on mixing as compared to the standard sampling in CMA-ES, and also finer granularity in the mixing. Either of these can have a positive or negative impact on the performance, depending on the problem. For example, granularity of the sets in mixing should ideally be tuned to the linkage between variables in the problem, if any. Too fine grained mixing may disrupt optimized substructures, while too coarse grained mixing may not achieve the optimal possible performance increase.

To our surprise, the non-restarting M-CMA-ES dominates the restarting M-CMA-ES on the first problem class for all parameter settings (the observed difference for 2 sets is not significant). Note that for single CMA-ES and multiple independent CMA-ES (# sets = 1), the restarting does contribute to achieve better performance, see [3] and Figure 3. On the Ł$_{100}$ problems, restarting M-CMA-ES dominates the non-restarting variant as expected – the observed differences are only significant for 4 sets and more.

On both problem classes, the non-restarting variant remains fairly consistent in performance across the range of tested parameter settings, with only few statistically significant differences, while restarting makes the mixing more sensitive to parameter settings.

From this experiment, we select the restarting M-CMA-ES with 2 sets for Ł$_4$ problems (even though the difference is statistically insignificant, the observed performance of the restarting variant is highest). For the Ł$_{100}$ problems, we select the restarting variant with 15 sets. Furthermore, we will also compare with independent parallel CMA-ES populations, to isolate the effect of having multiple populations, and the effect of the mixing mechanism. We select the restarting and non-restarting variants for the Ł$_4$ and Ł$_{100}$ problems respectively.

	Success		Evaluations	
	Ł$_4$	Ł$_{100}$	Ł$_4$	Ł$_{100}$
CMA-ES	94.24%	96.72%	297142	266974
10 CMA-ES	94.29%	97.44%	366655	178673
M-CMA-ES	95.06%	98.6%	478158	101670

Table 1: Success percentage after 10^7 evaluations, and average number evaluations in successful runs. All observed differences with the best are significant (Wilcoxon signed-rank test, confidence 95%).

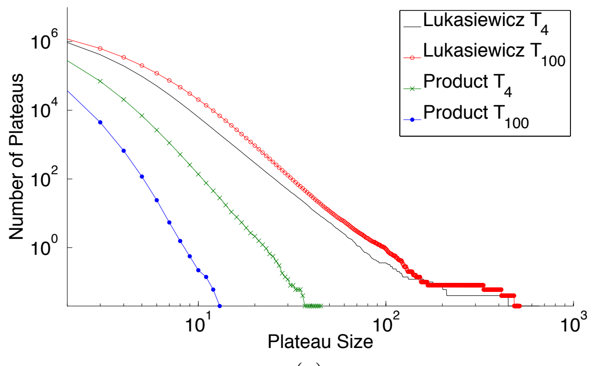
As we compare standard CMA-ES with multiple independent CMA-ES and M-CMA-ES, we use the same population sizes ($\mu = 2$, $\lambda = 20$) in each, meaning that the latter two algorithms effectively perform an order of magnitude more evaluations each generation. There is no need to compare these with the performance of CMA-ES with population size times 10, since we previously showed that increasing population size by this amount on these problems does not improve performance, given a budget of 10^7 function evaluations [3]. Figure 4 shows the run-length distributions of these three variants of CMA-ES, and Table 1 summarizes these results.

We observe that, in agreement with our hypothesis based on the landscape analysis, using mixing when solving the Ł$_{100}$ problems has more effect than on the Ł$_4$ problems. The performance on the more granular problems increases both in terms of probability of solving an instance, and average number of evaluations required to solve an instance. On the coarse-grained problems, there is an increase in success probability, but at the cost of a much higher average number of evaluations. Note that not only having multiple populations improves performance, but also the mixing mechanism in itself. Furthermore, the single CMA-ES starts solving instances almost an order of magnitude faster, simply because these easy problems do not require the extra populations, nor the mixing to be solved efficiently, and therefore the algorithms that have these mechanisms spend too much evaluations on redundant mechanisms.

6. OTHER LOGICS

Since our objective function formulation is essentially independent of the underlying logic and its operators, we will now briefly evaluate two new problem classes with a different conjunction operator. Similar to how we generated the Ł$_4$ and Ł$_{100}$ problems, we build two new problem classes, replacing the conjunction ($\otimes$) from the Łukasiewicz logic with that from the Product logic[2] ($[\alpha \otimes \beta]_{\mathcal{I}} = [\alpha]_{\mathcal{I}} \times [\beta]_{\mathcal{I}}$), again with bounds from $\mathbb{T}_4$ and $\mathbb{T}_{100}$. We refer to these problem classes as P_4 and P_{100}. Since the Product logic conjunction is a smooth function, we can expect a decrease in the number of plateaus as compared to the Ł$_k$ problems. This is confirmed in Figure 5 (a). The P_k problems have orders

[2]The mixing of operators from different logics is justified as the formulation of any problem as SAT$_\infty$ leaves open the choice of operators; their choice depends on the desired problem properties.



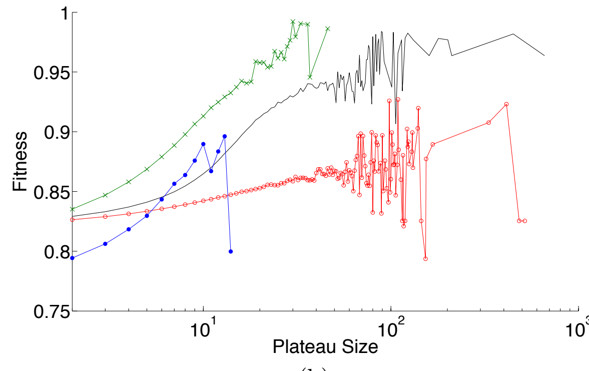
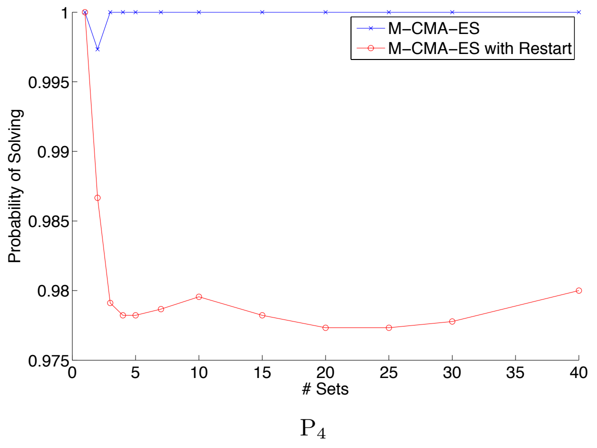


Figure 5: (a) The number of plateaus as a function of plateau size, as estimated by the number of random steps at the same fitness value. (b) The fitness of plateaus as a function of plateau size.



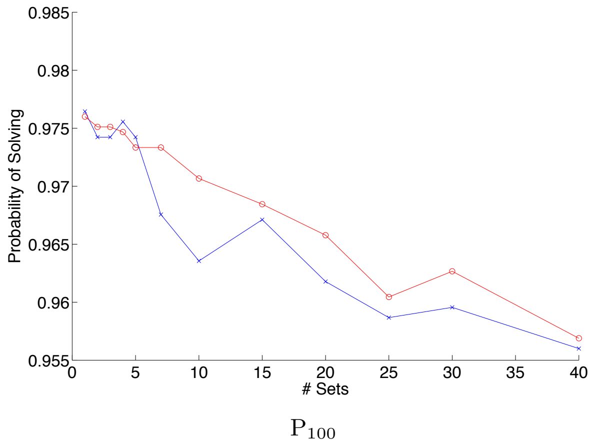
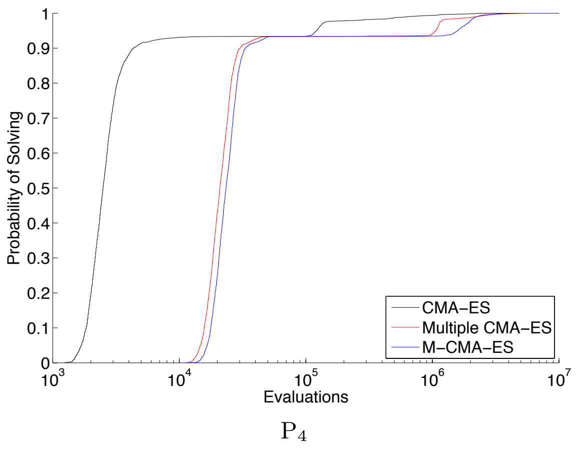


Figure 6: Success ratio achieved given a budget of 10^7 function evaluations as a function of the number of marginal product sets used in mixing on the Product problem classes.



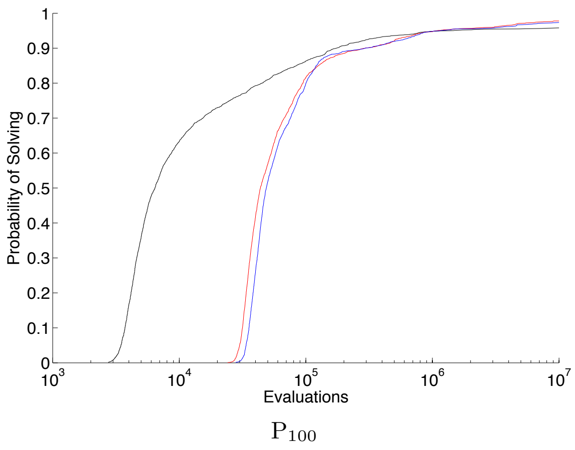


Figure 7: Run length distributions of CMA-ES, multiple CMA-ES and M-CMA-ES on the Product problem classes.

	Success		Evaluations	
	P_4	P_{100}	P_4	P_{100}
CMA-ES	100%	95.78%	26434	66056
10 CMA-ES	100%	97.78%	117490	186221
M-CMA-ES	100%	97.33%	149006	176690

Table 2: Success percentage after 10^7 evaluations, and average number evaluations in successful runs. All observed differences with the best are significant (Wilcoxon signed-rank test, confidence 95%). Note that in this case 100% means 45 instances solved, since in both classes, only 45 out of 50 instances are known to us to be satisfiable.

of magnitude less plateaus, and of much smaller size. Also, the average fitness of plateaus rises greatly with plateau size on both new classes (Figure 5 (b)). These two properties combined lead us to the expectation that the mixing mechanism will provide little to no improvement over standard CMA-ES.

As we will evaluate M-CMA-ES on these new problem classes, we again first optimize the parameter controlling the number of disjunct sets used in mixing. Figure 6 shows the results of this optimization, performed with the same experimental setup as in Section 5.1. Note that the relative performance of the restarting and non-restarting variant is very similar on these problem classes as on the first two problem classes. The non-restarting variant again dominates in performance on the first problem class, achieving much better performance, suggesting that the restart-mechanism, when combined with mixing, kicks in too soon and prevents the mixing to achieve its full potential. When no restarts are executed, mixing has the chance of trying many different mixes, since for each mix, variables are randomly distributed over the sets, and mixing partners are randomly selected.

We selected for both problem classes and both mixing and independent CMA-ES the non-restarting variant, M-CMA-ES with 2 and 4 sets on the two classes respectively. Figure 7 shows the run-length distributions of CMA-ES, multiple CMA-ES and M-CMA-ES on the P_k problem classes, and Table 2 summarises these results. It is clear that the mixing only worsens performance by spending precious evaluations on a mechanism that is not necessary. Having multiple CMA-ES populations makes a difference on the P_{100} problems in terms of probability of solving an instance (on the P_4 problems, success is already 100%), but the average number of evaluations is on both problem classes much higher than for a single CMA-ES population.

7. CONCLUSION

In this paper, we proposed an extension to CMA-ES, called M-CMA-ES, based on the landscape analysis of satisfiability problems in the Łukasiewicz fuzzy logic. M-CMA-ES incorporates two types of mechanisms: the standard CMA-ES model-based optimization at population level, and the recombination optimization at the meta-population level. The number of evaluations that go to either mechanism can be controlled and needs to be balanced for optimal effect. Product logic problems clearly benefit more from the standard CMA-ES optimization, and have no need for population recombination. Łukasiewicz problems on the other hand have many plateaus and benefit much from spending part of the evaluations on recombination. In future work, we will consider other mixing operators besides simple variable exchange, and will analyse their effect on inter-population diversity, as well as how this affects performance.

8. ACKNOWLEDGEMENT

This work was partially funded by a joint VUB-UGent Research Foundation-Flanders (FWO) project. Also, Tim Brys is funded by a Ph.D grant of the Research Foundation Flanders (FWO).

9. REFERENCES

[1] A. Auger and N. Hansen. Performance Evaluation of an Advanced Local Search Evolutionary Algorithm. In *IEEE Congress on Evolutionary Computation*, pages 1777–1784, 2005.

[2] T. Brys, Y.-M. De Hauwere, M. De Cock, and A. Nowé. Solving satisfiability in fuzzy logics with evolution strategies. In *Proceedings of the 31st Annual North American Fuzzy Information Processing Society Meeting*, 2012.

[3] T. Brys, M. M. Drugan, P. A. N. Bosman, M. De Cock, and A. Nowé. Local search and restart strategies for satisfiability solving in fuzzy logics. In *Proceedings of the IEEE Symposium Series on Computational Intelligence - SSCI-2013.* 2013.

[4] R. Hähnle. Many-valued logic and mixed integer programming. *Annals of Mathematics and Artificial Intelligence*, 12:231–263, 1994.

[5] P. Hájek. *Metamathematics of Fuzzy Logic.* Springer, 1998.

[6] N. Hansen. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[7] J. Janssen, S. Schockaert, D. Vermeir, and M. De Cock. Reducing fuzzy answer set programming to model finding in fuzzy logics. *CoRR*, 2011.

[8] N. Madrid and M. Ojeda-Aciego. Measuring inconsistency in fuzzy answer set semantics. In *Transactions on Fuzzy Systems*, volume 19, pages 605–622, 2011.

[9] S. Schockaert, M. De Cock, and E. E. Kerre. Spatial reasoning in a fuzzy region connection calculus. *Artificial Intelligence*, 173(2):258 – 298, 2009.

[10] S. Schockaert, J. Janssen, and D. Vermeir. Satisfiability checking in Łukasiewicz logic as finite constraint satisfaction. *Journal of Automated Reasoning*, 2012.

[11] U. Straccia and F. Bobillo. Mixed integer programming, general concept inclusions and fuzzy description logics. *Mathware & Soft Computing*, 2007.

[12] D. Thierens and P. A. N. Bosman. Optimal mixing evolutionary algorithms. In N. Krasnogor and P. L. Lanzi, editors, *GECCO*, pages 617–624. ACM, 2011.

[13] D. Van Nieuwenborgh, M. De Cock, and D. Vermeir. An introduction to fuzzy answer set programming. *Annals of Mathematics and Artificial Intelligence*, 50:363–388, 2007.