Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Stratification for Constraint-Based Multi-Objective Combinatorial Optimization

Miguel Terra-Neves, Inês Lynce, Vasco Manquinho
INESC-ID / Instituto Superior Técnico, Universidade de Lisboa, Portugal
{neves,ines,vmm}@sat.inesc-id.pt

Abstract

New constraint-based algorithms have been recently proposed to solve Multi-Objective Combinatorial Optimization (MOCO) problems. These new methods are based on Minimal Correction Subsets (MCSs) or P-minimal models and have shown to be successful at solving MOCO instances when the constraint set is hard to satisfy. However, if the constraints are easy to satisfy, constraint-based tools usually do not perform as well as stochastic methods. For solving such instances, algorithms should focus on dealing with the objective functions.

This paper proposes the integration of stratification techniques in constraint-based algorithms for MOCO. Moreover, it also shows how to diversify the stratification among the several objective criteria in order to better approximate the Pareto front of MOCO problems. An extensive experimental evaluation on publicly available MOCO instances shows that the new algorithm is competitive with stochastic methods and it is much more effective than existing constraint-based methods.

1 Introduction

In Multi-Objective Combinatorial Optimization (MOCO) applications such as scheduling [Iturriaga *et al.*, 2017] or green computing [Zheng *et al.*, 2016], there is more than one objective to be optimized, but there is no pre-defined hierarchy among the objective functions. Hence, there may exist multiple optimal solutions, known as Pareto optimal solutions, each of them favoring certain objectives at the expense of others. Several approaches have been proposed [Jackson *et al.*, 2009] that try to identify the Pareto front, i.e., all Pareto optimal solutions. However, this is known to be very hard for large MOCO instances, and most algorithms are just able to provide an approximation to the Pareto front.

There is a wide plethora of stochastic algorithms that try to approximate the Pareto front [Deb *et al.*, 2000; Xu and Fortes, 2010], and these methods are known for exhibiting a very good performance when the problem instances are not tightly constrained. However, their performance usually deteriorates when the constraint set becomes harder to satisfy. On the other hand, constraint-based methods are able to deal with large constraint sets, but have a hard time in integrating information from the several objectives to be optimized.

Lately, it has been shown that one can find the Pareto front of a MOCO instance by enumerating the set of Minimal Correction Subsets (MCSs) [Terra-Neves *et al.*, 2017] or the P-minimal models [Soh *et al.*, 2017] of a propositional formula. The main strength of these methods is on being able to quickly satisfy a large constraint set, but in several cases produce lower quality solutions than stochastic algorithms.

Constraint-based solvers for single objective problems are known to use different techniques on how to deal with the objective function. One example is to partition the terms in the objective function according to its weights. To this end, stratification techniques have been proposed [Ansótegui *et al.*, 2012]. In this work, we improve constraint-based MOCO algorithms by focusing on the optimization of several objective functions. In particular, we first propose a new solver that uses stratification in solving MOCO instances. Our main contributions are as follows: (1) formalization of the usage of stratification in MCS algorithms; (2) incorporation of stratification in solving MOCO instances; (3) a new stratification heuristic and (4) an extensive experimental evaluation on green computing MOCO instances that shows a huge improvement in performance enabled by stratification.

In this paper we start by defining MOCO and MCSs in section 2. Section 3 explains how to integrate stratification in MCS algorithms. Section 4 proposes the use of stratification in a constraint-based MOCO solver. Moreover, different stratification strategies are discussed and a new heuristic is also described. Experimental results showing the effectiveness of the newly proposed techniques are presented in section 5. Finally, the paper concludes in section 6.

2 Definitions

This section starts by describing Pseudo-Boolean Optimization (PBO) and Multi-Objective Combinatorial Optimization (MOCO). Next, Minimal Correction Subsets (MCSs) are defined. Finally, we review how MCSs can be used to find solutions for PBO and MOCO problems.

2.1 Multi-Objective Combinatorial Optimization

Let $X = \{x_1, \ldots, x_n\}$ be a set of n Boolean variables. A literal is either a variable x_i or its complement $\neg x_i$. Given a

x_1	x_2	x_3	$2x_1 + x_2$	$2\neg x_2 + 2x_3$
0	**1**	**1**	**1**	**2**
1	0	1	2	4
1	**1**	**0**	**3**	**0**
1	1	1	3	2

Table 1: Satisfiable assignments and respective costs for the instance in example 2.2.

set of m literals $l_1, \ldots, l_m$ and their respective coefficients $\omega_1, \ldots, \omega_m \in \mathbb{N}$, a Pseudo-Boolean (PB) expression is a weighted sum of literals $\sum \omega_i \cdot l_i$. Given an integer $k \in \mathbb{N}$, a linear PB constraint has the form:

$$\sum \omega_i \cdot l_i \bowtie k, \quad \bowtie \in \{\leq, \geq, =\}. \tag{1}$$

Consider a set $F = \{c_1, \ldots, c_m\}$ of m PB constraints and a cost function f defined over a set of X Boolean variables. The PBO [Boros and Hammer, 2002] problem consists of finding a complete assignment $\alpha : X \to \{0, 1\}$ that satisfies all constraints in F, denoted $\alpha(F) = 1$, and minimizes the value of f, denoted $f(\alpha)$. If F is unsatisfiable, then $\alpha(F) = 0$ for any assignment α. Analogously, given a PB constraint c, $\alpha(c) = 1$ ($\alpha(c) = 0$) denotes that α satisfies (does not satisfy) c.

Example 2.1. *Consider a PBO instance defined over $X = \{x_1, x_2, x_3\}$ where $F = \{(x_1 + x_2 + x_3 \geq 2)\}$ is the set of PB constraints and $f(X) = 4x_1 + 2x_2 + 3x_3$ is the cost function. In this case, $\alpha = \{(x_1, 0), (x_2, 1), (x_3, 1)\}$ is an optimal assignment with a cost of 5. All other assignments to X either do not satisfy F, or result in a higher value of $f(X)$.*

In this paper, the negation operator $\neg$ can also be applied to a given PB constraint c. In this case, $\neg c$ represents that c cannot be satisfied. Observe that $\neg c$ can be easily converted into an equivalent PB constraint by updating the relational operator and the right-hand side. Finally, note that, by definition, all coefficients are required to be positive, but constraints with negative coefficients can easily be converted to equivalent ones with all positive coefficients. The same can be applied for the cost function.

A MOCO [Ulungu and Teghem, 1994] instance is composed of two sets: a set $F = \{c_1, \ldots, c_m\}$ of constraints that must be satisfied and a set $O = \{f_1, \ldots, f_k\}$ of cost functions to minimize. In this work, we focus on the special case where $c_1, \ldots, c_m$ are PB constraints and $f_1, \ldots, f_k$ are PB expressions over a set X of Boolean variables.

Given two complete assignments α, α' such that $\alpha \neq \alpha'$ and $\alpha(F) = \alpha'(F) = 1$, we say that α dominates α', written $\alpha \prec \alpha'$, if, and only if, $\forall_{f \in O} f(\alpha) \leq f(\alpha')$ and $\exists_{f' \in O} f'(\alpha) < f'(\alpha')$. α is said to be Pareto optimal if, and only if, no other complete assignment α'' exists such that $\alpha''(F) = 1$ and $\alpha'' \prec \alpha$. In MOCO, the goal is to find the set of Pareto optimal solutions, also referred to as Pareto front.

Example 2.2. *Let $F = \{(x_1 + x_2 + x_3 \geq 2)\}$ be the set of constraints and $O = \{(2x_1 + x_2), (2\neg x_2 + 2x_3)\}$ the set of cost functions of a MOCO instance. The costs for each possible satisfiable assignment to F are shown in Table 1. The lines that correspond to Pareto optimal solutions are highlighted in bold, while the other are dominated. For example, $\{(x_1, 1), (x_2, 0), (x_3, 1)\}$ is not Pareto optimal because it is dominated by $\{(x_1, 0), (x_2, 1), (x_3, 1)\}$.*

For most problems, finding all Pareto optimal solutions in a reasonable amount of time is extremely hard. Hence, our focus is on finding the best approximation of the Pareto front.

2.2 Minimal Correction Subsets

Let F be an unsatisfiable set of PB constraints. A Minimal Correction Subset (MCS) of F is a subset $C \subseteq F$ such that $F \setminus C$ is satisfiable and C is minimal, i.e., $(F \setminus C) \cup \{c\}$ is unsatisfiable for all $c \in C$.

Example 2.3. *Consider the unsatisfiable set of PB constraints $F = \{(x_1 + x_2 = 1), (x_1 \geq 1), (x_2 \geq 1)\}$. F has three MCSs $C_1 = \{(x_1 \geq 1)\}$, $C_2 = \{(x_2 \geq 1)\}$ and $C_3 = \{(x_1 + x_2 = 1)\}$.*

There are several algorithms described in the literature for finding MCSs [Bailey and Stuckey, 2005; Felfernig *et al.*, 2012; Marques-Silva *et al.*, 2013; Mencía *et al.*, 2015]. Optionally, these algorithms can receive as argument a set of hard constraints F_H that must be satisfied and a set F_S of soft constraints for which we want to find an MCS $C \subseteq F_S$. In this case, C is an MCS if $F_H \cup (F_S \setminus C)$ is satisfiable and $F_H \cup (F_S \setminus C) \cup \{c\}$ is unsatisfiable for all $c \in C$. For simplicity, we assume that F_H is always satisfiable, but this can be checked using a single call to a satisfiability solver.

MCSs can be used to find approximate solutions of PBO instances. Let F be the set of constraints and $f(X) = \sum \omega_i \cdot l_i$ the cost function of a PBO instance. Let $L(f)$ be the set of all literals in f and $L^{\neg}(f)$ the set of clauses built from the negation of the literals in $L(f)$, i.e., $L^{\neg}(f) = \bigcup_{l_i \in L(f)} \{(\neg l_i)\}$. Applying an MCS algorithm with $F_H = F$ and $F_S = L^{\neg}(f)$, produces an MCS C of $L^{\neg}(f)$. We abuse notation and denote as $f(C)$ the cost of C, defined as:

$$f(C) = \sum_{(\neg l_i) \in C} \omega_i. \tag{2}$$

Any assignment that satisfies $F \cup L^{\neg}(f) \setminus C$ will have a cost of $f(C)$, which provides an approximation to the optimum of the PBO instance. Actually, the PBO problem can be reduced to finding the MCS $C \subseteq L^{\neg}(f)$ that minimizes $f(C)$ [Birnbaum and Lozinskii, 2003].

MCSs can also be used to find the Pareto front of MOCO instances. Let F and O be the constraint and cost function sets, respectively, of a MOCO and $L^{\neg}(O) = \bigcup_{f \in O} L^{\neg}(f)$. Terra-Neves *et al.* [2017] proved that one can find the Pareto front by enumerating all MCSs of $L^{\neg}(O)$.

Example 2.4. *Consider $F = \{(x_1 + x_2 + x_3 \geq 2)\}$ and $O = \{(2x_1 + x_2), (2\neg x_2 + 2x_3)\}$ from example 2.2. In this case, we have $L^{\neg}(O) = \{(\neg x_1), (\neg x_2), (x_2), (\neg x_3)\}$. As a result, there are three MCSs: $C_1 = \{(\neg x_2), (\neg x_1)\}$, $C_2 = \{(\neg x_2), (\neg x_3)\}$ and $C_3 = \{(\neg x_1), (x_2), (\neg x_3)\}$ with costs $(3, 0)$, $(1, 2)$ and $(2, 4)$, respectively. Observe that these MCSs include all the Pareto optimal assignments highlighted in table 1.*

3 Stratification

Stratification has been successfully applied to boost the performance of Maximum Satisfiability (MaxSAT)

Algorithm 1: Generic stratified MCS algorithm

Input: F_H, F_S
1 $(P_1, P_2, \ldots, P_k) \leftarrow \texttt{Partition}(F_S)$
2 $(S_1, C) \leftarrow (F_H, \emptyset)$
3 **for** $i \leftarrow 1$ **to** k **do**
4 $\quad C_i \leftarrow \texttt{MCS}(S_i, P_i)$
5 $\quad S_{i+1} \leftarrow S_i \cup (P_i \setminus C_i) \cup \bigcup_{c \in C_i} \{\neg c\}$
6 $\quad C \leftarrow C \cup C_i$
7 **return** C

solvers [Ansótegui *et al.*, 2012]. Its use has been suggested by Mencía *et al.* [2015] for approximating MaxSAT with MCSs, but stratified MCS enumeration has not been properly described in the literature yet. In this section, we describe a generic stratified approach that can be used to find lower cost MCSs faster with any MCS algorithm. Given a cost function $f(X) = \sum \omega_i \cdot l_i$, the main goal of stratification is to focus the search on satisfying the literals in $L^{\neg}(f)$ with larger coefficients. The procedure starts by partitioning $L^{\neg}(f)$ into k sets $P_1, P_2, \ldots, P_k$ such that all literals in P_i have larger coefficients than those in P_{i+1} for all $1 \leq i < k$. Next, an MCS algorithm is applied considering all hard constraints and just the literals in P_1 (the ones with larger coefficients) as soft. In the next iteration, P_2 is added to the set of soft constraints. At each iteration, a new partition is added and the procedure ends when all partitions have been considered.

The pseudo-code for the generic MCS stratified approach is presented in algorithm 1. It receives as input a set of hard constraints F_H and a set of soft constraints F_S and it returns an MCS C of F_S. First, it starts by partitioning F_S into a set of k partitions such that $F_S = \bigcup_{i=1}^{k} P_i$. At each iteration, S_i denotes the set of constraints that are to be satisfied. In the first iteration, all hard constraints are considered (line 2). On the other hand, C denotes the set of unsatisfied soft constraints. Then, for each partition P_i, it computes an MCS C_i of P_i with S_i as the set of hard constraints (line 4). Since the constraints in $P_i \setminus C_i$ are satisfied, these are added to S_{i+1} (line 5). Moreover, the negation of the constraints in C_i is also added to S_{i+1} to further restrict the search space in the next MCS call. This is safe since C_i is an MCS of P_i, so satisfying $P_i \setminus C_i$ entails that the constraints in C_i cannot be satisfied. Finally, the constraints in C_i are added to C (line 6). Next, we prove that, after all k iterations, C is an MCS of F_S.

Proposition 1. *Algorithm 1 returns an MCS* $C = \bigcup_{i=1}^{k} C_i$ *of* $F_S = \bigcup_{i=1}^{k} P_i$.

Proof. Suppose that C is not an MCS of F_S. The two possible scenarios are: (1) $F_H \cup (F_S \setminus C)$ is unsatisfiable, or (2) there exists $c \in C$ such that $F_H \cup (F_S \setminus C) \cup \{c\}$ is satisfiable. Consider the first scenario[1]. C_k is an MCS of P_k, considering S_k as hard constraints, so $S_k \cup (P_k \setminus C_k)$ is satisfiable. Note that, for any $1 \leq i, j \leq k$, $C_i \subseteq P_i$ and if $c \in C_i$ and $c \in P_j$, then $c \in C_j$ [Terra-Neves *et al.*, 2017]. Therefore, we have $S_k \cup (P_k \setminus C_k) = F_H \cup \bigcup_{j=1}^{k} (P_i \setminus C_i) = F_H \cup (F_S \setminus C)$, contradicting scenario (1). Now let us consider the second scenario. Let C_i be an MCS of P_i such that $c \in C_i$. Then, $S_i \cup (P_i \setminus C_i) \cup \{c\}$ is unsatisfiable. We have $S_i \cup (P_i \setminus C_i) \cup \{c\} = F_H \cup \bigcup_{j=1}^{i} (P_j \setminus C_j) \cup \{c\}$ that is a subset of $F_H \cup (F_S \setminus C) \cup \{c\}$, contradicting scenario (2). □

[1]Note that we can ignore the negation of the constraints added in line 5, since these are entailed by the remainder of the formula.

Algorithm 2: Stratified MCS-based MOCO algorithm

Input: $F, O = \{f_1, f_2, \ldots, f_l\}$
1 **for** $i \leftarrow 1$ **to** l **do**
2 $\quad (P_1^i, P_2^i, \ldots, P_{k_i}^i) \leftarrow \texttt{Partition}(L^{\neg}(f_i))$
3 $(P_1 \ldots P_p) \leftarrow \texttt{Combine}((P_1^1 \ldots P_{k_1}^1) \ldots (P_1^l \ldots P_{k_l}^l))$
4 $(S_1, C) \leftarrow (F, \emptyset)$
5 **for** $i \leftarrow 1$ **to** p **do**
6 $\quad C_i \leftarrow \texttt{MCS}(S_i, P_i)$
7 $\quad S_{i+1} \leftarrow S_i \cup (P_i \setminus C_i) \cup \bigcup_{c \in C_i} \{\neg c\}$
8 $\quad C \leftarrow C \cup C_i$
9 **return** C

Algorithm 1 computes a single MCS C, but it can be used to find another MCS. This can be done by adding a *blocking* constraint to F_H such that at least one constraint in C must be satisfied and re-executing the algorithm. Hence, algorithm 1 can be used to enumerate all MCSs of $\bigcup_{i=1}^{k} P_i$ by blocking previous MCSs in subsequent invocations of the algorithm.

4 Stratification for Multi-Objective Combinatorial Optimization

Stratification for MOCO is not straightforward for two reasons: (1) unlike the single-objective case, a good MOCO algorithm should not only converge fast but also produce an approximation with high diversity in regard to costs; (2) cost functions can be very different in nature. This section describes how stratification can be extended to MOCO in order to find better approximations of the Pareto front and faster.

Algorithm 2 presents the pseudo-code for the usage of stratification for solving MOCO instances. The algorithm has as input a constraint set F and a cost function set $O = \{f_1, \ldots, f_l\}$. The algorithm starts by using stratification for each cost function f_i (line 2). Hence, the set of soft clauses $L^{\neg}(f_i)$ is split into k_i partitions $P_1^i P_2^i \ldots P_{k_i}^i$. Next, a single partition sequence is built by mixing the partitions of each cost function (line 3), where we have $p = \sum_{i=1}^{l} k_i$. For example, assuming we have two cost functions f_1 and f_2 with partitions $P_1^1 P_2^1$ and $P_1^2 P_2^2 P_3^2$, a possible sequence could be $P_1^1 P_1^2 P_2^2 P_2^1 P_3^2$. Finally, the sequence of partitions is solved as in algorithm 1, generating an MCS of $\cup_{f_i \in O} L^{\neg}(f_i)$.

Recall that the Pareto front can be identified using MCS enumeration. Therefore, using algorithm 2 to enumerate all MCSs enables the generation of the Pareto front of a MOCO instance. However, for most pratical instances, generating the complete Pareto front is not feasible in a reasonable amount of time. Hence, our goal is to use this approach to find the best possible approximation of the Pareto front by generating a diverse set of good quality solutions in a given time limit.

In our algorithm, the partitioning of cost functions is performed separately, since the cost functions can be very different in nature. Observe that if we repeatedly use the same sequence for MCS enumeration, some cost functions can be favored at the expense of others. For example, if our partition sequence is $P_1^1 P_1^2 P_2^2 P_2^1 P_3^2$, algorithm 2 will focus on satisfying the high cost literals of $L^{\neg}(f_1)$ first, which might entail higher costs for f_2. Therefore, f_1 will be favored at the expense of f_2, which violates the diversity requirement. In order to promote diversity, the combination procedure (line 3) should generate a different partition sequence in each call to algorithm 2. In our implementation, partitions are mixed by selecting the next partition of a cost function with uniform probability at each iteration. For instance, assuming we are building a sequence and we already have $P_1^1 P_1^2$, the next partition would be P_2^1 or P_2^2, each having a probability of $\frac{1}{2}$ of being selected.

4.1 Partitioning Heuristics

The partitioning heuristic is an important component of the stratified approach and can have a significant impact on performance. In general, given a cost function f, we sort the literals in $L^{\neg}(f)$ in decreasing order of their respective coefficients. Then, for each literal $l \in L^{\neg}(f)$, we add l to the current partition P and check whether P is a good partition according to some criteria. If so, P is saved and the next partition is generated using the remaining literals.

Partitioning heuristics differ in the criteria for deciding if P is a good partition. One possibility is to simply split each cost function into a fixed number p of partitions. In the FIXED heuristic, P is considered a good partition if $p \cdot |P| \geq |L^{\neg}(f)|$. This heuristic is used in Microsoft's SMT solver Z3 [Bjørner *et al.*, 2015] for solving MaxSAT problems. In this paper we propose the Literal-Weight Ratio (LWR) heuristic that is a variant of the heuristic proposed by Ansótegui *et al.* [2012]. In LWR, P is a good partition if the ratio between the number of literals in P and the number of distinct coefficients among those literals is above a given threshold β, i.e., $\frac{|P|}{|W|} > \beta$, where W is the set of coefficients of the literals in P.

In both heuristics, our implementation does not produce partition sets where literals with the same weight are assigned to different partitions, i.e. we handle all literals with the same weight in the same step when generating new partitions.

4.2 Combining Stratification and Division Reduction

It was recently proved that, in the presence of a cost function f of the form $f(X) = \sum_{i=1}^{k} \frac{g_i(X)}{h_i(X)}$, f can be replaced by k cost function pairs g_i and $-h_i$, as long as $g_i(X) \geq 0$ and $h_i(X) > 0$ for all $1 \leq i \leq k$, without sacrificing Pareto-optimal solutions of the original problem [Terra-Neves *et al.*, 2018]. This technique is known as *division reduction* and the set of g_i and $-h_i$ functions is referred as *reduction product*.

Suppose that we have a MOCO with cost functions f_1 and f_2, where f_1 is a sum of k divisions and f_2 is a regular PB expression. The reduced MOCO will have $2k+1$ cost functions, where $2k$ of them are the reduction product of f_1. The stratification process, as described in the previous section, would assign a uniform probability of selection to each function, resulting in an overall probability of $\frac{2k}{2k+1}$ of selecting a partition of f_1, which is much larger than f_2's probability of $\frac{1}{2k+1}$. Therefore, stratification is very likely to favor f_1, violating the diversity requirement. To prevent this behavior, the partitioning process must be aware that some cost functions may be the product of division reduction.

We implemented two strategies for combining stratification and division reduction. The first strategy, referred to as MERGE, merges the $2k$ literal sets $L^{\neg}(g_i), L^{\neg}(-h_i)$ of the reduction product of f into a single set $L^{\neg}(f)$, i.e., $L^{\neg}(f) = \bigcup_{i=1}^{k} (L^{\neg}(g_i) \cup L^{\neg}(-h_i))$ Then, $L^{\neg}(f)$ is partitioned as described in section 4.1, with the exception that the coefficients in the reduction product are considered instead of the ones in f. The second strategy relies on adapting the selection probabilities of reduced cost functions instead of considering them as one. Recall the previous example with f_1 and f_2, where f_1 is a sum of k divisions. Initially, both have a selection probability of $\frac{1}{2}$. After the division reduction of f_1, we split f_1's probability uniformly among the $2k$ functions in its reduction product, resulting in a probability of $\frac{1}{2} \cdot \frac{1}{2k} = \frac{1}{4k}$ for each function, while f_2 maintains a probability of $\frac{1}{2}$. In what follows, we refer to this strategy as SPLIT.

5 Experimental Results

In this section, the performance of the stratified MCS approach for MOCO is evaluated on instances of the Virtual Machine Consolidation (VMC) problem. In VMC, we have several servers with fixed resource capacities and Virtual Machines (VMs) with requirements of those same resources. Each VM must be placed in some server, but server capacities cannot be exceeded and some VMs cannot be placed in the same server. There exists an initial placement, i.e., a VM can be associated with an initial server, incurring a migration cost if the VM is placed in a different one. A migration budget constraint can be used to enforce an upper limit on the migration costs, and is specified as a percentile bp of the total memory capacity of the servers. The goal is to find a placement for all VMs that satisfies the constraints and simultaneously minimizes (1) energy consumption in the data center, (2) migration costs and (3) resource wastage. The latter is a measure of the imbalance of server resource usage.

A detailed description of the VMC problem and the MOCO formulation, can be found in the literature [Terra-Neves *et al.*, 2017; Zheng *et al.*, 2016]. Note that the resource wastage cost function is a sum of divisions, which is handled using division reduction. The evaluation is performed on publicly available VMC benchmarks[2], based on subsets of workload traces randomly selected from the Google Cluster Data project[3].

In order to evaluate the quality of the approximations to the Pareto front, two quality indicators are used. The Hypervolume (HV) quality indicator [Zitzler and Thiele, 1999] provides a combined measure of convergence and diversity. Larger values of HV mean that the solution set is composed of solutions of better quality and/or diversity. The Inverted

[2] http://sat.inesc-id.pt/dome/

[3] http://code.google.com/p/googleclusterdata/

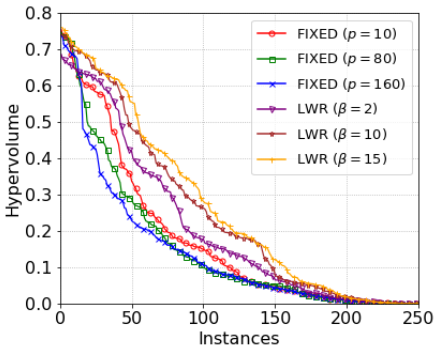

Figure 1: HV distributions for partitioning strategies ($bp = 100\%$).

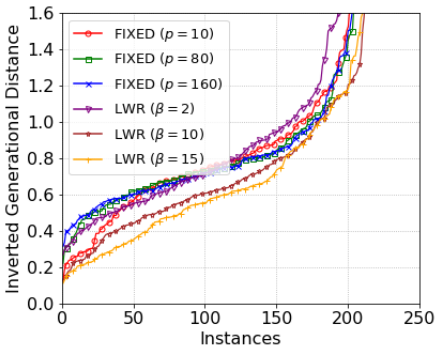

Figure 2: IGD distribution for partitioning strategies ($bp = 100\%$).

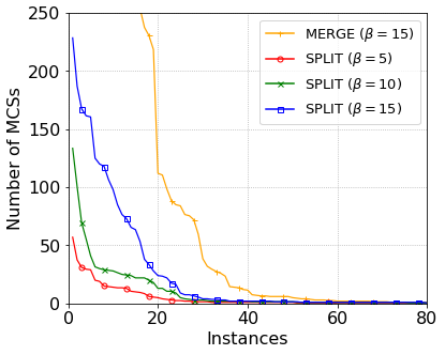

Figure 3: Distribution of number of MCSs for division handling ($bp = 100\%$).

Generational Distance (IGD) indicator [Zhang and Li, 2007] measures the average Euclidean distance, in the cost space, between the Pareto optimal solutions and the approximation returned by the algorithm, and a smaller value is preferred. Since the Pareto front is unknown, we use instead the combination of the solutions produced by all algorithms evaluated.

In our approach, any MCS algorithm can be used. However, in order to make a proper evaluation and comparison with the work of Terra-Neves et al. [2017], we also use the ClauseD (CLD) algorithm [Marques-Silva *et al.*, 2013] to compute MCSs. All algorithms are implemented in Java and Sat4j-PB v.2.3.4 [Le Berre and Parrain, 2010] is used as satisfiability checker. Moreover, we merge partitions whenever the satisfiability checker reaches a fixed number of conflicts (200000 in our experiments). This results from the fact that some calls might be too constrained. When this occurs, we relax the formula by adding the next partition. For example, if the satisfiability algorithm reaches the conflict limit when computing an MCS of P_i, the MCS algorithm stops, P_{i+1} is added and the search continues for an MCS of $P_i \cup P_{i+1}$. If all partitions have been added, there is no conflict limit.

Each algorithm was ran with a memory limit of 4 GB and a time limit of 1800 seconds. Randomized algorithms were executed with 10 different seeds for each instance, and the analysis is performed using the median values over all executions. Finally, it was observed that algorithms with better performance have lower standard deviation values than algorithms with worse performance. This occurred for both hypervolume and IGD indicators.

5.1 Partitioning Heuristics

First, we evaluate the impact of the partitioning heuristics (section 4.1) on the performance of algorithm 2. Figures 1 and 2 show the distributions of the HV and IGD values, respectively, obtained by the different heuristic configurations with a migration budget of 100%. A point (x, y) in the HV (IGD) distribution plot indicates that the given configuration obtained an HV (IGD) equal to or greater (lower) than y for x instances. For example, the point $(50, 0.56)$ on line 'LWR ($\beta = 15$)' in figure 1 indicates that, using the LWR heuristic with $\beta = 15$, the algorithm obtained HVs equal to or greater than 0.56 for 50 instances.

Different configuration values were tried for the LWR and FIXED heuristics. In particular, LWR was configured with values of $\beta \in \{2, 5, 10, 15, 20, 30\}$ and FIXED with $p \in \{10, 20, 40, 80, 160\}$. In order to improve readability, results are shown solely for representative values. However, all best configurations are included. Additionally, in figure 2, IGD values larger than 1.6 are not displayed. This is done also for readability of the IGD plots.

Observe that LWR is strictly better than FIXED both in terms of HV and IGD, except when $\beta = 2$ in the IGD plot. However, with $\beta = 2$ was the worst configuration of LWR. For all other values of β, the LWR performance is very similar. The same also occurs for different configurations of the FIXED heuristic. Hence, it was observed that both heuristics are robust, and that the partitioning strategy itself has a much higher impact in performance than parameter fine tuning.

5.2 Handling Division Reduction

This section evaluates the MERGE and SPLIT strategies proposed in section 4.2 for handling the division reduction of the wastage optimization criteria. Both were run using the LWR partitioning heuristic with different values of β.

MERGE and SPLIT showed very similar IGD and HV distributions, i.e. they produced approximations of very similar quality of the Pareto front. However, the number of MCSs found was very different. Therefore, we choose to show the number of MCSs found for representative configurations in figure 3. Note that the number of MCSs is a relevant metric since it can be seen as a measure of progress. The more MCSs an algorithm is able to find, the closer it is to proving that the Pareto front was found instead of an approximation. Figure 3 clearly shows that MERGE is able to find many more MCSs than SPLIT. At best, SPLIT with $\beta = 15$ is able to find close to 230 MCSs for one instance, while MERGE is able to find at least as much for close to 20 instances. Actually, MERGE is able to find close to 500 MCSs for some instances.

5.3 Comparison with State-of-the-Art

In this section, we compare SCLD, our new stratified approach for MOCO, with the simple CLD enumeration algorithm and the MCSEnumPD variant that incorporates an MCS diversification technique [Terra-Neves *et al.*, 2018]. We also compare with the VMPMBBO [Zheng *et al.*, 2016] algorithm for VMC and with the general purpose evolutionary

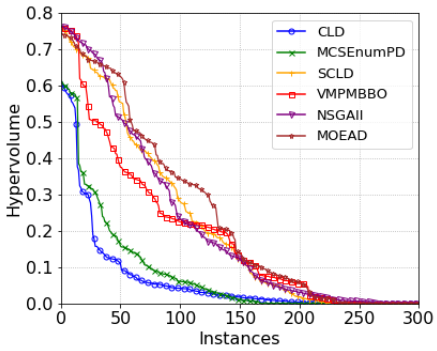

Figure 4: HV distributions ($bp = 100\%$).

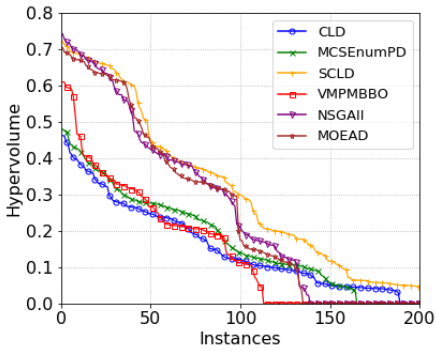

Figure 6: HV distributions ($bp = 5\%$).

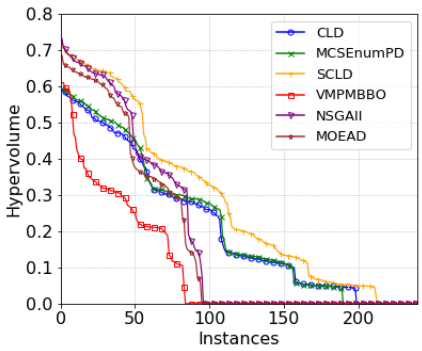

Figure 8: HV distributions ($bp = 1\%$).

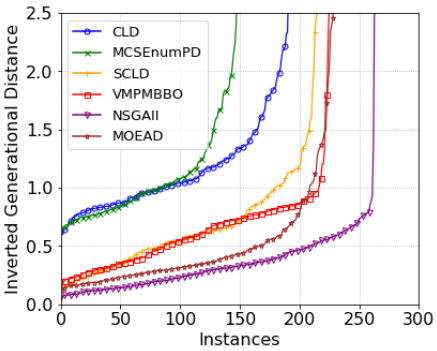

Figure 5: IGD distributions ($bp = 100\%$).

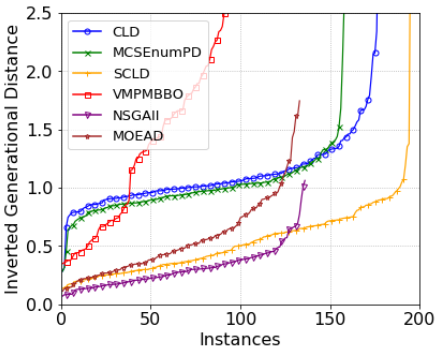

Figure 7: IGD distributions ($bp = 5\%$).

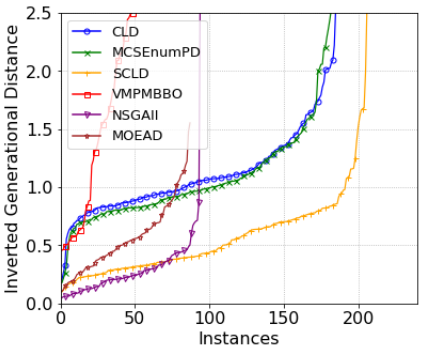

Figure 9: IGD distributions ($bp = 1\%$).

algorithms NSGAII [Deb *et al.*, 2000] and MOEAD [Zhang and Li, 2007]. SCLD was configured with the LWR partitioning heuristic ($\beta = 15$) and the MERGE division reduction strategy. MCSEnumPD, VMPMBBO and NSGAII were configured as suggested in the literature [Zheng *et al.*, 2016; Terra-Neves *et al.*, 2017; 2018]. The only difference is in how we encoded the individuals in NSGAII's population. In this work, the regular integer encoding is used instead of the binary integer encoding, since it produces better results on the VMC problem. Further details on encodings for evolutionary algorithms can be found in the literature [Rothlauf, 2006]. MOEAD was configured with crossover and mutation rates of 0.8 and 0.05 respectively, a population size of 100, a neighborhood size of 20, a 0.9 probability of crossover with individuals from the neighborhood and a maximum of 2 individuals that can be replaced by a single offspring.

Figures 4 and 5 show the HV and IGD distributions of each algorithm for VMC instances with $bp = 100\%$. Observe that the new techniques proposed in this paper allow SCLD to obtain a huge performance improvement when compared with previous state-of-the-art constraint-based algorithms for MOCO, CLD and MCSEnumPD. Moreover, considering the hypervolume indicator, SCLD is competitive with stochastic algorithms NSGAII, MOEAD and VMPMBBO for these instances. Nevertheless, stochastic algorithms are still able to obtain better results measured by IGD. These results are indicative that NSGAII is able to converge faster, but SCLD finds approximations with higher diversity.

Figures 6, 7, 8 and 9 present the results when migration budgets of VMC instances are constrained to 5% and 1%. Observe that NSGAII's, MOEAD's and VMPMBBO's performance degrades considerably as the budget decreases because these algorithms have a much harder time dealing with more tightly constrained instances. On the other hand, constraint-based methods for MOCO thrive in such scenarios. Nevertheless, note that SCLD is still able to improve considerably on both CLD and MCSEnumPD, thus confirming that our approach still improves previous algorithms even on tighter instances. Overall, SCLD is the first constraint-based algorithm to be competitive with stochastic approaches for the VMC instances where $bp = 100\%$, and is the best performing algorithm when $bp \leq 5\%$.

6 Conclusion and Future Work

Interest has been surging in solving MOCO formulations using constraint-based methods. Until recently, the state of the art was mostly populated by stochastic methods, which are able to find good approximations fast, but are unsuitable for constrained problems. On the other hand, constraint-based methods are able to quickly satisfy tightly constrained problems, but have a harder time producing good quality approximations. This paper proposes the use of stratification as a means to guide MCS-based MOCO algorithms towards good quality solutions more effectively. Experimental results in a large set of MOCO instances show that stratification provides a very significant performance improvement for MCS algorithms, being now able to compete with (and often outperform) stochastic methods in terms of approximation quality.

As future work, we propose to improve the diversity of the approximation by dynamically adapting the cost function selection probabilities, and to exploit these in order to account

for user preferences [Deb and Sundar, 2006]. Moreover, recent work in high-school timetabling problems has shown that significant improvements can be obtained by combining stochastic and constraint-based methods [Demirovic and Musliu, 2017]. Such a combined algorithm for MOCO could reap the benefits of both worlds with none of the weaknesses.

Acknowledgments

This work was supported by national funds through FCT with references UID/CEC/50021/2013 and SFRH/BD/111471/2015.

References

[Ansótegui *et al.*, 2012] C. Ansótegui, M. L. Bonet, J. Gabàs, and J. Levy. Improving SAT-Based Weighted MaxSAT Solvers. In *International Conference on Principles and Practice of Constraint Programming*, pages 86–101, 2012.

[Bailey and Stuckey, 2005] J. Bailey and P. Stuckey. Discovery of Minimal Unsatisfiable Subsets of Constraints Using Hitting Set Dualization. In *Symposium on Practical Aspects of Declarative Languages*, pages 174–186, 2005.

[Birnbaum and Lozinskii, 2003] E. Birnbaum and E. Lozinskii. Consistent Subsets of Inconsistent Systems: Structure and Behaviour. *Journal of Experimental & Theoretical Artificial Intelligence*, 15(1):25–46, 2003.

[Bjørner *et al.*, 2015] N. Bjørner, A.-D. Phan, and L. Fleckenstein. νZ - An Optimizing SMT Solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 194–199, 2015.

[Boros and Hammer, 2002] E. Boros and P. L. Hammer. Pseudo-Boolean Optimization. *Discrete Applied Mathematics*, 123(1):155–225, 2002.

[Deb and Sundar, 2006] K. Deb and J. Sundar. Reference Point Based Multi-Objective Optimization Using Evolutionary Algorithms. In *Genetic and Evolutionary Computation Conference*, pages 635–642, 2006.

[Deb *et al.*, 2000] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimisation: NSGA-II. In *International Conference on Parallel Problem Solving from Nature*, pages 849–858. Springer, 2000.

[Demirovic and Musliu, 2017] E. Demirovic and N. Musliu. MaxSAT-based Large Neighborhood Search for High School Timetabling. *Journal of Computers & Operations Research*, 78:172–180, 2017.

[Felfernig *et al.*, 2012] A. Felfernig, M. Schubert, and C. Zehentner. An Efficient Diagnosis Algorithm for Inconsistent Constraint Sets. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 26(1):53–62, 2012.

[Iturriaga *et al.*, 2017] S. Iturriaga, B. Dorronsoro, and S. Nesmachnow. Multiobjective Evolutionary Algorithms for Energy and Service Level Scheduling in a Federation of Distributed Datacenters. *International Transactions in Operational Research*, 24(1-2):199–228, 2017.

[Jackson *et al.*, 2009] D. Jackson, H. Estler, D. Rayside, et al. The Guided Improvement Algorithm for Exact, General-Purpose, Many-Objective Combinatorial Optimization. Technical Report MIT-CSAIL-TR-2009-033, MIT, 2009.

[Le Berre and Parrain, 2010] D. Le Berre and A. Parrain. The Sat4j Library, Release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7(2-3):59–6, 2010.

[Marques-Silva *et al.*, 2013] J. Marques-Silva, F. Heras, M. Janota, A. Previti, and A. Belov. On Computing Minimal Correction Subsets. In *International Joint Conference on Artificial Intelligence*, pages 615–622, 2013.

[Mencía *et al.*, 2015] C. Mencía, A. Previti, and J. Marques-Silva. Literal-Based MCS Extraction. In *International Joint Conference on Artificial Intelligence*, pages 1973–1979, 2015.

[Rothlauf, 2006] Franz Rothlauf. *Representations for Genetic and Evolutionary Algorithms (2.ed)*. Springer, 2006.

[Soh *et al.*, 2017] T. Soh, M. Banbara, N. Tamura, and D. Le Berre. Solving Multiobjective Discrete Optimization Problems with Propositional Minimal Model Generation. In *International Conference on Principles and Practice of Constraint Programming*, pages 596–614. Springer, 2017.

[Terra-Neves *et al.*, 2017] M. Terra-Neves, I. Lynce, and V. Manquinho. Introducing Pareto Minimal Correction Subsets. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 195–211. Springer, 2017.

[Terra-Neves *et al.*, 2018] M. Terra-Neves, I. Lynce, and V. Manquinho. Enhancing Constraint-Based Multi-Objective Combinatorial Optimization. In *AAAI Conference on Artificial Intelligence*. AAAI Press, 2018.

[Ulungu and Teghem, 1994] E. L. Ulungu and J. Teghem. Multi-Objective Combinatorial Optimization Problems: A Survey. *Journal of Multi-Criteria Decision Analysis*, 3(2):83–104, 1994.

[Xu and Fortes, 2010] J. Xu and J. Fortes. Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments. In *International Conference on Green Computing and Communications, & International Conference on Cyber, Physical and Social Computing*, pages 179–188. IEEE, 2010.

[Zhang and Li, 2007] Q. Zhang and H. Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[Zheng *et al.*, 2016] Q. Zheng, R. Li, X. Li, N. Shah, J. Zhang, F. Tian, K.-M. Chao, and J. Li. Virtual Machine Consolidated Placement Based on Multi-Objective Biogeography-Based Optimization. *Future Generation Computer Systems*, 54, 2016.

[Zitzler and Thiele, 1999] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.