Constraint Optimization Techniques for MultiObjective Branch and Bound Search

Emma Rollon[1] and Javier Larrosa[1]

Universitat Politècnica de Catalunya
Jordi Girona 1-3, Edificio Omega — 08034 Barcelona, Spain
erollon@lsi.upc.edu, larrosa@lsi.upc.edu

1 Introduction

In *Constraint Optimization Problems* (COP) the task is to find the best solution according to some preferences expressed by means of cost functions [1]. *Branch and Bound* (BB) [2] is an exact general search algorithm for COPs solving. BB is the most usual algorithm in the mono-objective case. The efficiency of BB depends on its hability to detect dead-ends, that is, nodes that do not have any solution below. Dead-end detection is done with a heuristic function that computes an underestimation or *lower bound* of the current subproblem. In recent years, many heuristic functions have been proposed. For instance, all *weighted* CSP local consistencies [3, 4] can be used for this purpose. Another alternative is *Mini-bucket elimination* (MBE) [5]. MBE is a generic inference method well-known in *Constraint Programming* [6]. It computes a lower bound of the optimal solution (assuming minimization). Therefore, MBE is usually used inside BB to improve its pruning capability.

Many real world problems involve multiple measures of performance, or objectives, which should be optimized simultaneously. The simultaneous optimization of multiple, possibly competing, objective functions deviates from single function optimization in that it seldom admits a single, perfect solution. Instead, multiobjective constraint optimization problems tend to be characterized by a family of alternatives which must be considered equivalent in the absence of information concerning the relevance of each objective relative to the others.

In *MultiObjective Constraint Optimization Problems* the task is to find the *efficient frontier*, that is, the set of equivalent or non-dominated costs of the set of feasible solutions. *MultiObjective Branch and Bound* search (MO-BB) has not been widely studied in the multiobjective context [7]. One reason is the lack of general approximation algorithms to compute lower bounds. In our recent work [8], we have extended MBE from mono-objective to multiobjective optimization problems, yielding *MultiObjective Mini-Bucket Elimination* (MO-MBE). MO-MBE computes a *lower bound set* [9] of the efficient

frontier. As a consequence, MO-MBE can be used as a heuristic function in a multiobjective branch and bound algorithm. In this paper we describe how MO-MBE can be combined with multiobjective branch and bound search. The resulting algorithm is a simple, extremely generic, exact multiobjective solving method. Our experiments on *bi-objective combinatorial auctions* and *bi-objective weighted vertex cover* problems demonstrates the performance of the new approach.

The structure of this paper is as follows: Section 2 provides some preliminaries on multiobjective optimization. Section 3 introduces MultiObjective Mini-Bucket Elimination. Section 4 describes the multiobjective extension of a generic branch and bound algorithm and shows how MO-MBE can be integrated as a heuristic function. Section 5 reports some experimental results. Finally, Section 6 gives some conclusions and points out some directions of future work.

2 Preliminaries

Let $\mathcal{X} = (x_1, \dots, x_n)$ be an ordered set of variables and $\mathcal{D} = (D_1, \dots, D_n)$ an ordered set of domains. Domain D_i is a finite set of potential values for x_i. We call d the largest domain size. The assignment (i.e, instantiation) of variable x_i with $a \in D_i$ is noted $(x_i := a)$. A *tuple* is an ordered set of assignments to different variables $(x_{i_1} := a_{i_1}, \dots, x_{i_k} := a_{i_k})$. The set of variables $(x_{i_1}, \dots, x_{i_k})$ assigned by a tuple t, noted $var(t)$, is called its *scope*. The size of $var(t)$ is the *arity* of t. When the scope is clear by the context, we omit the variables and express the tuple as a sequence of domain values $(a_{i_1} \dots a_{i_k})$. We focus on two basic operations over tuples: the *projection* of t over $A \subseteq var(t)$, noted $t[A]$, is a sub-tuple of t containing only the instantiation of variables in A. Let t and s be two tuples having the same instantiations to the common variables. Their *join*, noted $t \cdot s$, is a new tuple which contains the assignments of both t and s. Projecting a tuple t over the empty set $t[\emptyset]$ produces the empty tuple λ. We say that a tuple t is a *complete instantiation* when $var(t) = \mathcal{X}$. Sometimes, when we want to emphasize that a tuple is a complete instantiation we will call it X.

Let consider problems with one objective. A *weighted CSP* (WCSP) [10] is a tuple $P = (\mathcal{X}, \mathcal{D}, \mathcal{F}, \top)$, where $\mathcal{X}$ and $\mathcal{D}$ are variables and domains. $\mathcal{F} = \{f_1, \dots, f_r\}$ is a set of cost functions. Each cost function f_i is defined over $Y_i \subseteq \mathcal{X}$, called its *scope*. f_i associates costs (i.e., numbers) to tuples t such that $var(t) = Y_i$. We make the usual assumption of costs being natural numbers. $\top$ bounds the maximum acceptable cost of solutions. The objective function is, $F(X) = \sum_{i=1}^{r} f_i(Y_i)$. A *solution* is a complete assignment X such that $F(X) < \top$. An *optimal solution* is a solution X such that $\forall X',\ F(X) \leq F(X')$. The *optimum* of the objective function is the value $F(X)$. The task in a WCSP is to find the optimum and one (of the possibly many) optimal solutions X.

Let consider problems with p objectives. $\top = (\top_1, \ldots, \top_p)$ is a vector where each $\top_j \in \mathbb{N}$ is the maximum acceptable cost for the objective j. A p-vector $\mathbf{v} = (v_1, \ldots, v_p)$ is a vector of p components where each $v_j \in \mathbb{N}$ and $v_j \leq \top_j$. Let $\mathbf{v}$ and $\mathbf{u}$ be two *distinct* p-vectors. $\mathbf{v}$ *dominates* $\mathbf{u}$ (noted $\mathbf{v} < \mathbf{u}$) if $\forall j,\ v_j \leq u_j$. The sum of p-vectors is defined as,

$$\mathbf{v} + \mathbf{u} = \begin{cases} \top & \exists j,\ v_j + u_j \geq \top_j \\ (v_1 + u_1, \ldots, v_p + u_p) & \textit{otherwise.} \end{cases}$$

Let S be a set of p-vectors. We define its *non-domination closure* as $\langle S \rangle = \{\mathbf{v} \in S \mid \forall \mathbf{u} \in S,\ \mathbf{u} \not< \mathbf{v}\}$. Let S_1 and S_2 be two sets closed under non-domination. We say that S_1 *dominates* S_2 (noted $S_1 < S_2$) if $\forall \mathbf{v} \in S_2,\ \exists \mathbf{u} \in S_1\ s.t\ \mathbf{u} < \mathbf{v}$. A p-function f is defined over a set of variables $Y \subseteq \mathcal{X}$ such that $f(Y)$ is a p-vector. Let $x_i \in Y$ and $a \in D_i$, the partial instantiation of f with $x_i := a$, noted $f^{x_i:=a}$, is the new function obtained from f in which x_i has been *fixed* to a. Note that when x_i is the only variable of f, its instantiation produces a constant p-vector.

A *multiobjective weighted constraint satisfaction problem* (MO-WCSP) is defined as $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{F}, \top)$, where $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $\mathcal{D} = \{D_1, \ldots, D_n\}$ are variables and domains. $\top$ contains, for each objective, the maximum acceptable cost. $\mathcal{F}$ is a set of p-functions that define the multiobjective function $F(X) = \sum_{f \in \mathcal{F}} f(X)$. Given a complete assignment X, we say that it is *consistent* iff $F(X) \neq \{\top\}$. For clarity reasons, we consider that all the objective functions are additive. However, the same ideas posed in the MO-WCSP framework can be used for modelling problems where the objective functions are, for example, multiplicative (i.e. Probabilistic Frameworks [11]), or a combination of both. A *solution* is a consistent complete assignment. In the constraint programming context, the usual task is to find an optimal solution. A solution X is *efficient* or *Pareto optimal* if there is no better solution (i.e., $\forall X',\ F(X') \not< F(X)$). $\mathcal{X}_E$ is the set of efficient solutions and $\mathcal{EF}$ is the corresponding *efficient frontier*. The task in a MO-WCSP is to compute $\mathcal{EF}$ (and, possibly, one or all efficient solutions for each of its elements).

3 Mini-Bucket Elimination

MultiObjective Mini-Bucket Elimination (MO-MBE) [8] is the extension of the well-known mono-objective approximation algorithm *Mini-Bucket Elimination* (MBE) [5] to the multiobjective context. MO-MBE is a generic approximation algorithm that can be used to bound the efficient frontier when the problem is too difficult to be solved exactly. Assuming minimization problems, MO-MBE provides a lower bound set of the efficient frontier.

In the following, we augment p-functions by letting them to return a non-dominated *set* of p-vectors. MO-MBE uses two operations over p-functions: The *sum* of two p-functions f and g, noted $f + g$, is a new p-function that returns for each tuple the sum of the corresponding p-vectors, previous removal

f:

x_1	x_2	
a	a	$\{(3,2),(2,8)\}$
a	b	$\{(4,10)\}$
b	a	$\{\top\}$
b	b	$\{\top\}$

g:

x_2	x_3	
a	a	$\{(1,2)\}$
a	b	$\{(2,1)\}$
b	a	$\{(6,2),(11,1)\}$
b	b	$\{\top\}$

$f+g$:

x_1	x_2	x_3	
a	a	a	$\{(4,4),(3,10)\}$
a	a	b	$\{(5,3),(4,9)\}$
a	b	a	$\{(10,12)\}$
a	b	b	$\{\top\}$
b	a	a	$\{\top\}$
b	a	b	$\{\top\}$
b	b	a	$\{\top\}$
b	b	b	$\{\top\}$

$(f+g)\downarrow x_3$:

x_1	x_2	
a	a	$\{(4,4),(3,10),(5,3)\}$
a	b	$\{(10,12)\}$
b	a	$\{\top\}$
b	b	$\{\top\}$

Fig. 1. Sum and projection over 2-functions. $\top = (15, 18)$.

of dominated ones. The *elimination* of variable x_i from p-function f, noted $f \downarrow x_i$, is a new p-function not mentioning x_i that returns for each tuple the best p-vectors with respect to the eliminated variable. Formally, let f and g be two p-functions:

- Their sum $h = f + g$ is defined as,

$$h(t) = \langle \{\mathbf{v} |\ \ t = t' \cdot t'', \mathbf{v} = \mathbf{v}' + \mathbf{v}'', \mathbf{v}' \in f(t'), \mathbf{v}'' \in g(t'')\} \rangle$$

- The elimination of x_i, $h = f \downarrow x_i$ is defined as,

$$h(t) = \langle \{\mathbf{v} \mid \forall a \in D_i,\ \mathbf{v} \in f(t \cdot (x_i := a))\} \rangle$$

Consider as an example the 2-functions f and g in Figure 1 with $\top = (15, 18)$ under domains $\{a, b\}$. The sum $f+g$ is a 2-function $(f+g)(x_1, x_2, x_3)$. Note that in $(f+g)(a, b, a)$, the sum of the 2-vectors $(4, 10)$ and $(11, 1)$ is $\top$. As $\top$ is dominated by $(10, 12)$, it has been removed. The elimination of variable x_3 from $f+g$ is a 2-function $(f+g) \downarrow x_3(x_1, x_2)$. Note that in $(f+g) \downarrow x_3(a, a)$, the 2-vector $(4, 9)$ has been removed as a consequence of the non-domination closure. Moreover, $\top$ has also been removed from $(f + g) \downarrow x_3(a, b)$ for the same reason.

MO-MBE (Figure 2) has a control parameter z. It processes the problem eliminating variables one by one. For each variable x_i, the algorithm computes the so called *bucket* of x_i (line 2), noted $\mathcal{B}_i$, which contains all p-functions in $\mathcal{F}$ having x_i in its scope. Ideally, a new p-function would be computed by summing all functions in $\mathcal{B}_i$ and subsequently eliminating x_i. Since this is very space consumming, the bucket is partitioned into so-called *mini-buckets* (line 3). Each mini-bucket contains p-functions such that they do not jointly mention more than $z + 1$ variables. In each mini-bucket the functions are summed and subsequently x_i is eliminated (line 4). Then, $\mathcal{F}$ is updated by removing the functions in $\mathcal{B}_i$ and adding each g_{i_k} (line 5). After the last elimination, only an empty-scope p-function (i.e., a non-dominated set of p-vectors) remains. It contains a lower bound set of the original problem (line 7).

function MO-MBE($\mathcal{X}, \mathcal{D}, \mathcal{F}, \top, z$)
1. **for each** $i = n \dots 1$ **do**
2. $\mathcal{B}_i := \{h \in \mathcal{F} \mid x_i \in var(h)\}$;
3. $\{\mathcal{P}_{i_1}, \dots, \mathcal{P}_{i_r}\}$:=Partition($z, \mathcal{B}_i$);
4. **for each** $k = 1..r$ **do** $g_{i_k} := (\sum_{f \in \mathcal{P}_{i_k}} f) \downarrow x_i$;
5. $\mathcal{F} := (\mathcal{F} \cup \{g_{i_1}, \dots, g_{i_r}\}) - \mathcal{B}_i$;
6. **endfor**
7. **return** g_1

endfunction

Fig. 2. Description of MO-MBE. The input is a MO-WCSP instance $(\mathcal{X}, \mathcal{D}, \mathcal{F}, \top)$. The output is g_1, a zero-arity p-function which contains a lower bound set of the efficient frontier.

Note that, if MO-MBE returns $\{\top\}$ the problem does not have any solution. In general, greater values of z increment the number of p-functions included in each mini-bucket. Therefore, the lower bound set will be presumable closer to the efficient frontier. However, greater values of z produce higher arity functions which require more resources (i.e., space and time).

Theorem 1. *[8] MO-MBE with accuracy parameter z is space* $O(e \times \prod_{j=1}^{p-1} \top_j \times d^{z-1})$ *and time* $O(e \times \prod_{j=1}^{p-1} \top_j^2 \times d^z)$*, where e is the number of p-functions,* $\top_j$ *is the bound of objective j, p is the number of objectives, and d is the largest domain size.*

4 Depth First Branch and Bound

MultiObjective Branch-and-Bound (MO-BB) is a recursive description of a generic search schema for MO-WCSP solving. It searches depth-first the tree defined by the problem. During search, MO-BB maintains a set of non-dominated p-vectors corresponding to the best solutions found so far. In the minimization case, those vectors are an *upper bound set* or *top* of the optimal solution. When a new solution is found, its costs are added to the top and the non-dominated ones are retained as new top. Moreover, for each partial assignment, the algorithm computes a *lower bound set* using a *bounding evaluation function*, that is, an underestimation of its efficient frontier that can be obtained in the remaining problem. If the lower bound set is dominated by the top, the current path cannot lead to better solutions and the current branch can be pruned. As a result, the algorithm backtracks to a previous node.

In its description, MO-BB search (Figure 3) receives a set of variables $\mathcal{X}$ and the set of its feasible values $\mathcal{D}$, a set of p-functions $\mathcal{F}$, a top vector $\top$ and a non-dominated set $\mathcal{EF}$. After an initial call MO-BB $((x_1, \dots, x_n), (D_1, \dots, D_n)$,

procedure MO-BB($\mathcal{X}, \mathcal{D}, \mathcal{F}, \top, \mathcal{EF}$)
1. **if** $\mathcal{X} = \emptyset$ **then** $\mathcal{EF} := \langle \mathcal{EF} \cup \mathcal{F} \rangle$;
2. **else**
3. $x_i := Select(\mathcal{X})$;
4. **for each** $a \in D_i$ **do**
5. $\mathcal{F}' := \{f^{x_i:=a} \mid f \in \mathcal{F}\}$;
6. **if** $\mathcal{EF} \not< LB(\mathcal{X} - \{x_i\}, \mathcal{D} - \{D_i\}, \mathcal{F}', \top)$ **then**
7. MO-BB $(\mathcal{X} - \{x_i\}, \mathcal{D} - \{D_i\}, \mathcal{F}', \top, \mathcal{EF})$;
8. **endif**
9. **endfor**
10. **endif**

endprocedure

Fig. 3. Multi-Objective Depth-First Branch and Bound for optimization task.

$(f_1, \ldots, f_e), \top, \mathcal{EF} = \{\top\})$, the algorithm returns the efficient frontier of the problem in $\mathcal{EF}$. During search, the current efficient frontier is kept in $\mathcal{EF}$. When no variable remains, the current assignment is one of the best solutions found so far, so the efficient frontier is updated (line 1). Note that when there is no more variable to assign, $\mathcal{F}$ contains an empty scope p-function, that is, a constant p-function containing the optimal p-vectors of the current assignment. Then, the algorithm adds the p-vectors in $\mathcal{EF}$ and $\mathcal{F}$ and closed them under non-domination. When $\mathcal{X}$ is not empty, a variable is selected (line 3) and the algorithm sequentially attempts the assignment of its values to the p-functions in $\mathcal{F}$ (line 4-5). A lower bound set [9] of the cost of the current assignment is computed in the bounding evaluation function LB and compared with the current efficient frontier (line 6). If the current assignment may be extended, the search procedure proceeds by making a recursive call (line 7). Otherwise, the algorithm is in a dead-end and backtracks.

The performance of the search algorithm can be increased by reducing the explored search space. This reduction greatly depends on the *bounding evaluation function*. Therefore, the wisdom of the evaluation function to foresee a dead-end as soon as possible is a key factor in the branch and bound algorithm. MO-MBE can be executed inside branch and bound as a bounding evaluation function in order to provide lower bound sets of every subproblem. As MO-MBE is executed in each node, the control parameter z allows us to trade time for accuracy. In one hand, greater values of z will result in tighter lower bound sets. Therefore, the pruning capability of the algorithm will increase. However, the execution time will also increase. On the other, lower values of z will result in less tighter lower bound sets. However, the execution time will decrease and, as a consequence, reduce the time spent in every node.

5 Experimental Results

We have tested our approach in two different domains: *biobjective combinatorial auctions* and *biobjective weighted vertex cover* problems. The purpose of the experiments is to evaluate the performance of MO-BB using MO-MBE as an heuristic evaluation function (i.e. MO-BB$_{MOMBE}$) for solving MO-WCSP problems. To that end, we compare MO-BB$_{MOMBE}$ with the ϵ-constraint approach [7] based on search. Regarding MO-BB$_{MOMBE}$, experiments in the mono-objective case show that low values of the control parameter z usually provide reasonable good lower bounds with a very low cost [12]. Therefore, we follow the same criteria and set the control parameter $z = 2$ in all the experiments. For the ϵ-constraint approach, we use the well-known IlogSolver 6.1 as a solver engine. Moreover, the time spent for finding the ideal and nadir point that defines lower and upper bounds on the objective values of efficient solutions is not taken into account.

The time limit in all our experiments is 300 seconds. The execution time for unsolved instances is considered as that time limit. Therefore, for each domain, we report not just the cpu time, but also the percentage of solved instances within the time limit. We run all the experiments on a Pentium IV at 3GHz with $2GB$ of memory, running Linux.

5.1 Biobjective Combinatorial Auctions

Combinatorial auctions (CA) allow bidders to bid for indivisible subsets of goods [13]. In risk-conscious auctions, the auctioneer wants to control the risk of not being paid after a bid has been accepted, because it may cause large losses in revenue [14]. Consider a set of goods $\{1, 2, \ldots, n\}$ that go on auction. There are m bids. Bid j is defined by the subset of requested goods $X_j \subseteq \{1, 2, \ldots, n\}$, the money offer b_j and the probability of failure r_j. The auctioneer must decide which bids are to be accepted. If two bids have goods in common, only one of them can be accepted. The first objective is to maximize the auctioneer profit. The second objective is to minimize risk. Assuming independence, after a logarithmic transformation of probabilities, this objective can also be expressed as an additive function.

We have generated mono-objective CA using the PATH model of CATS generator [13] and randomly added payment failure probabilities to the bids in the range 0.0 to 0.3. We experiment on instances with 20 and 50 goods, varying the number of bids from 80 to 150. For each parameter configuration we generate samples of size 25.

Figure 4 reports the results obtained for instances with 20 and 50 goods corresponding to the plots on the right and on the left, respectively. MO-BB$_{MOMBE}$ outperformes ϵ-constraint in both configurations. For instances with 20 goods, MO-BB$_{MOMBE}$ solves all the instances within the time limit. However, ϵ-constraint only solves completely instances with 80 bids. Moreover, the solved percentage of ϵ-constraint decreases as the number of bids increases

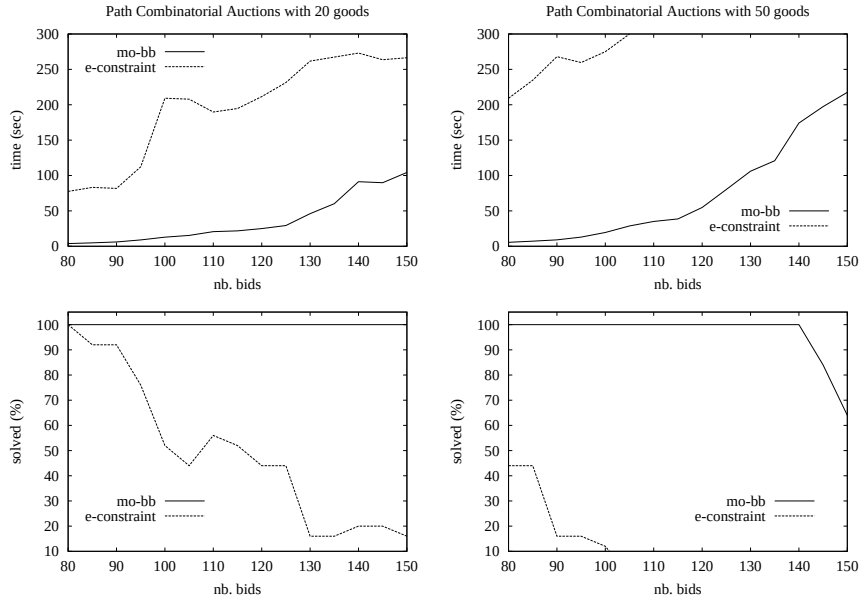

Fig. 4. Experimental results on bi-objective CA for 20 and 50 goods, respectively. Path distribution. Time limit 300 seconds.

and it is quite low (20%) from 130 bids. It is important to note that, as the time for unsolved instances is set to 300 seconds, its effect in the mean cpu time is minimized. For instances with 50 goods, MO-BB$_{MOMBE}$ does not solve 4 instances with 145 bids and 9 with 150 bids. However, it is important to note that those instances can be solved in less than 400 seconds. ϵ-constraint does not solve completely any parameter configuration. Moreover, it fails in solving all instances from 105 bids.

5.2 Biobjective Weighted Vertex Cover

Given a graph $G = (V, E)$, a *vertex cover* is a subset of vertices $S \subseteq V$ such that $\forall (u, v) \in E$, either $u \in S$ or $v \in S$. The *minimum vertex cover* is a vertex cover of minimum size. In the *weighted* version every vertex u has an associated weight $w(u)$ and the *weighted minimum vertex cover* is a vertex cover S with minimum $F(S) = \sum_{u \in S} w(u)$. In the biobjective version each vertex u has two weights $w_1(u)$ and $w_2(u)$ and the task is to minimize the two associated objective functions. In our experiments we generated random graph instances with parameters (N, E, C) where N is the number of vertices, E is the number of edges and C is the maximum weight. Instances are generated by randomly selecting E edges. For each vertex, two costs are randomly generated from the interval $[0 \dots C]$.

N (nb. vars)	E (nb. edges)	MO-BB$_{MOMBE}$ time (sec.)	%	ϵ-constraint time (sec.)	%
60	95	0.92	100	155.82	80
70	95	1.68	100	289.75	8
80	95	3.17	100	300	0
90	95	6.72	100	288	4
60	250	1.92	100	28.22	100
70	250	4.56	100	221.94	40
80	250	9.23	100	280.46	8
90	250	20.04	100	300	0
60	500	2.03	100	2.56	100
70	500	5.87	100	26.63	100
80	500	17.12	100	216.21	52
90	500	42.35	100	300	0
60	950	1.51	100	0.27	100
70	950	3.87	100	2.65	100
80	950	10.49	100	16.09	100
90	950	32.46	100	122.51	100

Fig. 5. Experimental results on biobjective weighted minimum vertex cover problems. Parameter C is set to 4. Mean values on 25 instances for each parameter configuration. Time limit 300 seconds.

We tested on samples of size 25 for the following parameter configurations $(\{60, 70, 80, 90\}, \{95, 250, 500, 950\}, 4)$. Figure 5 reports the results obtained. The first and second column show the number of variables and edges, respectively. The third and fourth columns report the mean cpu time and the percentage of solved instances within the time limit using MO-BB$_{MOMBE}$. The fifth and sixth column report the same information for ϵ-constraint approach. The first thing to be observed is that MO-BB$_{MOMBE}$ is clearly superior for all parameter configurations. MO-BB$_{MOMBE}$ solves all instances within the time limit. However, ϵ-constraint is only able to solve completely instances with 950 edges. When we fix the number of constraints and increase the number of variables, the efficiency of both approaches decreases. When fixing the number of variables and increasing the number of constraints, the behaviour of both approaches differs. Regarding MO-BB$_{MOMBE}$, the solving time increases until instances with 500 edges. For instances with 950 edges the time diminishes. However, the solving time for ϵ-constraint always decreases.

6 Conclusions and Future Work

MultiObjective Branch and Bound (MO-BB) is a general search schema for multiobjective constraint optimization problems. The search space is represented as a tree. The algorithm searches depth-first the tree defined by the problem. Its output is the efficient frontier of the problem. The efficiency of the algorithm greatly depends on its pruning ability which, in turn, depends on the computation of a good lower bound set at each visited node.

MultiObjective Mini-Bucket Elimination (MO-MBE) is an approximation algorithm for multiobjective constraint optimization problems. It has a control

parameter z which allow us to trade time and space for accuracy. Its output is a lower bound set of the efficient frontier of the problem. Therefore, it can be executed inside branch and bound as a bounding evaluation function in order to provide a lower bound set of every subproblem. We demonstrate the effectiveness of MO-BB using MO-MBE as a bounding evaluation function (i.e., MO-BB$_{MOMBE}$) in biobjective *combinatorial auctions* and *vertex cover* problems.

In our future work we want to evaluate the performance improvement of MO-BB$_{MOMBE}$ when using an initial good approximation of the efficient frontier. That initial approximation can be computed using approximate algorithms to compute upper bounds [7]. Moreover, we want to continue investigating the symbiosis between constraint programming and multiobjective optimization.

References

1. Bistarelli, S., Fargier, H., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G.: Semiring-based CSPs and valued CSPs: Frameworks, properties and comparison. Constraints **4** (1999) 199–240
2. Freuder, E., Wallace, R.: Partial constraint satisfaction. Artificial Intelligence **58** (1992) 21–70
3. Larrosa, J., Schiex, T.: In the quest of the best form of local consistency for weighted csp. In: Proc. of the 18^{th} IJCAI, Acapulco, Mexico (2003)
4. de Givry, S., Heras, F., Larrosa, J., Zytnicki, M.: Existential arc consistency: getting closer to full arc consistency in weighted csps. In: Proc. of the 19^{th} IJCAI, Edinburgh, U.K. (2005)
5. Dechter, R., Rish, I.: Mini-buckets: A general scheme for bounded inference. Journal of the ACM **50** (2003) 107–153
6. Dechter, R.: Constraint Processing. Morgan Kaufmann, San Francisco (2003)
7. Ehrgott, M., Gandibleux, X.: Multiple Criteria Optimization. State of the Art. Annotated Bibliographic Surveys. Kluwer Academic Publishers (2002)
8. Rollon, E., Larrosa, J.: Bucket elimination for multiobjective optimization problems. Journal of Heuristics **12** (2006) 307–328
9. Ehrgott, M., Gandibleux, X.: Bounds and bound sets for biobjective combinatorial optimization problems. Lecture Notes in Economics and Mathematical Systems **507** (2001) 241–253
10. Rossi, F., van Beek, P., Walsh, T.: 9. In: Handbook of Constraint Programming. Elsevier (2006)
11. Fargier, H., Lang, J.: Uncertainty in constraint satisfaction problems: a probabilistic approach. In: ECSQARU. (1993)
12. Kask, K., Dechter, R.: A general scheme for automatic generation of search heuristics from specification dependencies. Artificial Intelligence **129** (2001) 91–131
13. K.Leuton-Brown, M., Y.Shoham: Towards a universal test suite for combinatorial auction algorithms. ACM E-Commerce (2000) 66–76
14. Holland, A.: Risk Management for Combinatorial Auctions. PhD thesis, Dept. of Computer Science, UCC, Ireland. (2005)