Multi-Objective Russian Doll Search

Emma Rollon and **Javier Larrosa** *
Universitat Politecnica de Catalunya
Jordi Girona 1-3, 08034 Barcelona, Spain
erollon@lsi.upc.edu, larrosa@lsi.upc.edu

Abstract

Russian Doll Search (RDS) is a well-known algorithm for combinatorial optimization. In this paper we extend it from mono-objective to multi-objective optimization. We demonstrate its practical applicability in the challenging multiple-orbit SPOT5 instances. Besides being much more efficient than any other alternatives, multi-objective RDS can solve an instance which could not have been solved previously.

Introduction

Many real world problems involve multiple measures of performance or objectives, which should be simultaneously optimized (Ehrgott & Gandibleux 2002). Consider the scheduling of an earth observation satellite. The satellite orbits the earth while taking photographs requested by different customers. Each photograph has an importance. Moreover, there is a limitation on the on-board recording capacity. Therefore, the schedule should maximaze the overall importance of selected photographs while minimizing the overall memory usage. Optimality in *multi-objective* optimization is characterized by a family of pareto-optimal alternatives which are both optimal and incomparable among them. *Constraint Programming* deals with algorithms for *combinatorial problems* (Apt 2003). Most algorithmic research in the field is devoted to so-called satisfaction problems: problems that can be specified as a set of constraints that must be simultaneously satisfied. There is also a well established line of research on algorithms for mono-objective optimization (Rossi, van Beek, & Walsh 2006). Only very recently, some authors have started to look into multi-objective problems under the constraint programming paradigm (Junker 2006; Rollon & Larrosa 2006a; 2006b). The motivation of these works is not only to accommodate multi-objective optimization into the field, but also to exploit the multi-objective nature of some satisfaction problems. (Junker 2006; Rollon & Larrosa 2006b) say that a problem has multi-objective nature if it contains several constraints of the form $F_i(X) < K$. Solutions to such problems must be sufficiently good with respect each F_i, which is somehow related to the computation of pareto-optimal solutions.

*This work was funded with project TIN2006-15387-C03-02.

In this paper, we extend *Russian Doll Search* (RDS) (Verfaillie, Lemaître, & Schiex 1996), a well-known algorithm, from mono-objective to multi-objective optimization. This method has been proved to be very efficient in certain classes of important problems. In particular, it is the best current option in the SPOT5 benchmark (Bensana *et al.* 1996) which contains satellite scheduling instances. However, RDS fails when considering the so-called multiple orbit instances. We show that it is convenient to look at these instances as if they were multi-objective because they can be broken into independent subproblems. Then, we show that our multi-objective RDS is the best current alternative to deal with such subproblems. In particular, we solve for the first time instance 1504.

Preliminaries

Let $X = (x_1, \ldots, x_n)$ be an ordered set of *variables* and $\mathcal{D} = (D_1, \ldots, D_n)$ an ordered set of *domains*, where D_i is the finite set of potential values for x_i. The *assignment* (i.e, instantiation) of variable x_i with $a \in D_i$ is noted $(x_i = a)$. A *complete assignment* is an assignment of all the variables and it is noted A. We will use X_i^j as a short hand for $x_i, x_{i+1}, \ldots, x_j$, a subset of the variables. Similarly, A_i^j will denote $(x_i = a_i), (x_{i+1} = a_{i+1}), \ldots, (x_j = a_j)$, an assignment of X_i^j (note that $A = A_1^n$). The *join* of two assignments is noted $A_i^j \cdot A_k^l$.

A *cost function* f with scope $var(f) \subseteq X$ associates a value to each assignment of $var(f)$. We will use $|f|$ as a short-hand of $|var(f)|$, which is the *arity* of f. Let $x_i \in var(f)$ 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 *conditioned* (i.e., fixed) to a. Note that, when x_i is the only variable in the scope of f, its instantiation produces a constant.

Mono-objective WCSP

A *Weighted CSP* (WCSP) (Rossi, van Beek, & Walsh 2006) is a triple $P = (X, \mathcal{D}, \mathcal{F})$, where X and $\mathcal{D}$ are variables and domains. $\mathcal{F} = \{f_1, \ldots, f_r\}$ is a set of cost functions.The objective function is,

$$F(X) = \sum_{i=1}^{r} f_i(Y_i)$$

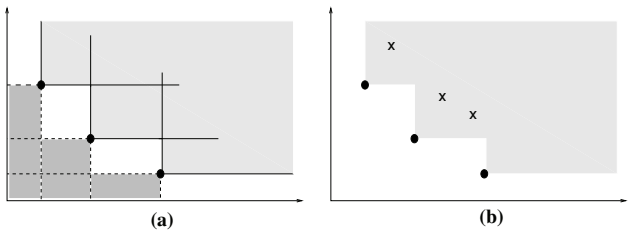

Figure 1: Efficient frontiers.

where $Y_i \subseteq X$ is the scope of f_i. We make the usual assumption of costs being natural numbers. A *solution* is a complete assignment A. An *optimal solution* is a complete assignment A such that $\forall A',\ F(A) \leq F(A')$. The *optimum* of the objective function is the value $F(A)$. The usual task of interest is to find the optimum and one (of the possibly many) optimal assignments A.

Consider a WCSP $P = (X, \mathcal{D}, \mathcal{F})$ and an assignment A_i^j. Problem P *conditioned* to A_i^j, noted $P(A_i^j)$, is a new problem in which functions in $\mathcal{F}$ are partially assigned with A_i^j. Formally, $P(A_i^j) = (X - X_i^j, \mathcal{D} - \{D_i, \ldots, D_j\}, \{f(A_i^j) | f \in \mathcal{F}\})$.

A WCSP can be represented by its *interaction graph* $G = (V, E)$ (also called *constraint graph*), which contains one node per variable and one arc connecting any pair of nodes whose variables are included in the scope of some function. When the constraint graph is not connected, it means that the problem is formed by independent subproblems. As a consequence, each part can be solved independently. The optimal cost of the overall WCSP is the sum of the cost obtained for each connected part.

Multi-objective Weighted CSP

Let consider problems with p objectives. A *cost vector* $\vec{v} = (v_1, \ldots, v_p)$ is a vector of p components where each $v_j \in \mathbb{N}$ represents the cost with respect objective j. A *multi-objective function* f is a function that associates a cost vector to each assignment of its scope. Let $\vec{v}$ and $\vec{u}$ be two cost vectors. We say that $\vec{v}$ *dominates* $\vec{u}$ (noted $\vec{v} \leq \vec{u}$) if $\forall j,\ v_j \leq u_j$. We say that $\vec{v} < \vec{u}$ if $\vec{v} \leq \vec{u}$ and $\vec{v} \neq \vec{u}$. The sum of cost vectors is the usual point-wise sum.

Let α be a set of cost vectors. We define its *non-domination closure* as $\|\alpha\| = \{\vec{u} \in \alpha\ | \forall \vec{v} \in \alpha,\ \vec{v} \not< \vec{u}\}$. A set of vectors closed under non-domination is called *frontier*. Let α and β be two frontiers. We say that α *dominates* β (noted $\alpha \leq \beta$) if $\forall \vec{v} \in \beta,\ \exists \vec{u} \in \alpha\ s.t\ \vec{u} \leq \vec{v}$. We say that $\alpha < \beta$ if $\alpha \leq \beta$ and $\alpha \neq \beta$. The sum of frontiers is $\alpha + \beta = \|\{\vec{w} = \vec{u} + \vec{v} |\ \vec{u} \in \alpha, \vec{v} \in \beta\}\|$

Example 1 *Consider bi-dimensional cost vectors. Figure 1 (a) depicts a set α of vectors (each vector is a dot in the 2D space). Each vector $\vec{v}$ is the top-right corner of a rectangle (doted lines). Any point in the rectangle would dominate $\vec{v}$. Each $\vec{v}$ is also the bottom-left corner of an infinitely large rectangle (solid lines). Any point in that rectangle would be dominated by $\vec{v}$. The set α is a frontier since no point is dominated by any other. Figure 1 (b) depicts two frontiers α as dots and β as crosses. As it can be seen, $\alpha < \beta$ because the area dominated by α contains all the elements of β.*

A *multi-objective weighted constraint satisfaction problem* (MO-WCSP) (Rollon & Larrosa 2006a) is a triple $\mathcal{P} = (X, \mathcal{D}, \mathcal{F})$, where X, $\mathcal{D}$ and $\mathcal{F}$ are variables, domains and multi-objective functions, respectively. The sum of functions in $\mathcal{F}$ defines the objective function F. The only difference with respect to mono-objective WCSP is that multi-objective functions are used. A *solution* is a complete assignment A. A solution A is *efficient* or *pareto optimal* if $\forall A',\ F(A') \not< F(A)$. The *efficient frontier* is the set $\mathcal{E} = \{F(A) | \forall A',\ F(A') \not< F(A)\}$. The usual task of interest is to compute $\mathcal{E}$ and, for each $\vec{v} \in \mathcal{E}$, one (of the possibly many) pareto optimal assignment with cost $\vec{v}$.[1]

As in the WCSP case, each MO-WCSP instance has an associated interaction graph G computed in the same way: one node per variable and one arc connecting any pair of nodes whose variables are included in the scope of some mo-function. When the graph is not connected, each subpart can be solved independently. The efficient frontier of the overall problem is the sum of the efficient frontier of each subpart.

Depth First Branch and Bound

Multi-objective problems can be solved with a depth-first branch-and-bound schema, which searches depth-first the tree of possible assignments. The output of the search is the efficient frontier of the problem $\mathcal{E}$. The algorithm handles two frontiers, that we call *lower* and *upper bound frontiers*. The *upper bound frontier* (`ubf`) is an overestimation of the efficient frontier $\mathcal{E}$. Formally, a *valid* `ubf` satisfies that $\mathcal{E} \leq$ `ubf`.

During search, `ubf` is the set of cost vectors of the best solutions found so far (which are the current candidates to be part of the efficient frontier). `ubf` is a frontier because when a new cost vector is added, all those that are dominated can be removed (they cannot be part of $\mathcal{E}$). It is easy to see that during the execution of the algorithm `ubf` is valid. When the whole search tree is traversed, clearly `ubf` $= \mathcal{E}$. Moreover, at each visited node, the algorithm computes a *lower bound frontier* (`lbf`) of the current problem using a *bounding evaluation function*. `lbf` is an underestimation of the efficient frontier of the current problem. Formally, a *valid* `lbf` satisfies that, `lbf` $\leq \mathcal{E}'$ where $\mathcal{E}'$ is the efficient frontier of the original problem *conditioned* with the current assignment. The algorithm backtracks if the upper bound dominates the lower bound (i.e., `ubf` $\leq$ `lbf`), because it implies that the global efficient frontier dominates the current subproblem efficient frontier (i.e., $\mathcal{E} \leq \mathcal{E}'$). Consequently, the current assignment cannot lead to any new efficient cost vector.

MO-BB (Figure 2, lines 1-11) is a recursive description of the previous algorithm. MO-BB receives a MO-WCSP instance $P = (X, \mathcal{D}, \mathcal{F})$ and the current best frontier `ubf`. In the initial call, P is the original problem and `ubf` can be set to any frontier such that $\mathcal{E} \leq$ `ubf`. First of all, if no variable remains, the cost of the current assignment (stored

[1] Sometimes $\mathcal{E}$ is too large to be reported and must be approximated. In this paper we disregard such possibility.

procedure MO-BB($P = (X, \mathcal{D}, \mathcal{F}), \texttt{ubf}$)
1. **if** $X = \emptyset$ **then** $\texttt{ubf} := \|\texttt{ubf} \cup \sum_{f \in \mathcal{F}} f\|$;
2. **else**
3. $x_j := Select(X)$;
4. **for each** $a \in D_j$ **do**
5. $\mathcal{F}' := \{f(x_j = a) \mid f \in \mathcal{F}\}$;
6. $X' := X - \{x_j\}$; $\mathcal{D}' := \mathcal{D} - \{D_j\}$;
7. **if** $\texttt{ubf} \not\leq \text{LBF}(X', \mathcal{D}', \mathcal{F}')$ **then**
8. MO-BB $(P' = (X', \mathcal{D}', \mathcal{F}'), \texttt{ubf})$;
9. **endif**
10. **endfor**
11. **endif**

endprocedure

function MO-RDS(P) **return** frontier
12. $\texttt{rds}[n+1] := \{\vec{0}\}$;
13. **for each** i **from** n **downto** 1 **do**
14. $\texttt{rds}[i]$:=UBF($\texttt{rds}[i+1]$);
15. MO-BB($P_i, \texttt{rds}[i]$);
16. **endfor**
17. **return** $\texttt{rds}[1]$;

endfunction

function SMO-RDS(P) **return** frontier
18. $\texttt{rds}[n+1] := \{\vec{0}\}$;
19. **for each** i **from** n **downto** 1 **do**
20. **for each** $a \in D_i$ **do**
21. $\texttt{rds}[i, a]$:=UBF($\texttt{rds}[i+1], a$);
22. MO-BB($P_{ia}, \texttt{rds}[i, a]$);
23. **endfor**
24. $\texttt{rds}[i] := \|\bigcup_{a \in D_i} \texttt{rds}[i, a]\|$
25. **endfor**
26. **return** $\texttt{rds}[1]$;

endfunction

Figure 2: Multi-objective algorithms.

in $\mathcal{F}$ as a set of completely assigned functions) is included in ubf and dominated vectors are removed (line 1). If X is not empty, an arbitrary unassigned variable x_j is selected[2] (line 3). Then, the algorithm sequentially attempts the assignment of its values to the cost functions in $\mathcal{F}$ (line 4-6). A lower bound frontier of the cost of the current assignment is computed with the bounding evaluation function LBF and compared with ubf (line 7). If the backtracking condition does not hold, the search procedure proceeds by making a recursive call (line 8). Otherwise, the algorithm detects a dead-end and backtracks.

The performance of the search algorithm depends on the quality of the *upper* and *lower bound frontiers*, because tight bounds anticipate dead-ends and prune at early stages of the search. The simplest ubf is the empty set. Better approximations can be obtained, for example, with local search heuristics (Ehrgott & Gandibleux 2002). Let A_1^{i-1} be the current partial assignment and $P(A_1^{i-1})$ be the current subproblem. The simplest lbf is a singleton frontier made by the sum of cost vectors from completely instantiated cost functions,

$$\texttt{lbf}_s = \sum_{\substack{f \in P(A_1^{i-1}) \\ |f|=0}} f$$

Borrowing an idea of (Freuder & Wallace 1992) in the mono-objective context, the previous lower bound can be improved. Let $x_k,\ i \leq k \leq n$ be an unassigned variable and $a \in D_k$ one of its domain values,

$$ic_{ka} = \sum_{\substack{f \in P(A_1^{i-1}) \\ var(f)=\{x_k\}}} f(x_k = a)$$

is the cost of the singleton assignment $(x_k = a)$, and $ic_k = \|\bigcup_{a \in D_k} ic_{ka}\|$ is the set of the best incomparable alternatives that can be obtained if x_k is assigned to any of its domain values. The following expression is a valid lower bound frontier,

$$\texttt{lbf} = \texttt{lbf}_s + \sum_{k=i}^{n} ic_k$$

In the next section, we describe multi-objective *Russian Doll Search* which computes more accurate lower and upper bound frontiers.

Multi-objective Russian Doll Search

Russian Doll Search (RDS) (Verfaillie, Lemaître, & Schiex 1996) is a branch-and-bound algorithm which invests in high quality upper and lower bounds. It was originally proposed for mono-objective problems. The idea of RDS is to replace one search by n successive searches on nested subproblems, where n is the number of variables in the problem. The first subproblem involves only one variable. Each successive subproblem results from adding one variable to the previous one. Finally, the last subproblem is the whole problem. The key of the algorithm is that the optimal assignments of all the solved subproblems are systematically recorded in order to help future searches. In particular, each search can use the optimal cost of previous subproblems as a lower bound of the correponding partial assignment. Moreover, each search uses the extension of the optimal assignment of the previous subproblem to improve its upper bound. In the following, we generalize it to the multi-objective context.

Let $P = (X, \mathcal{D}, \mathcal{F})$ be a MO-WCSP. As in the original mono-objective paper we assume that $\mathcal{F}$ does not contain unary cost functions. Otherwise, the definition of ic_{ka} needs to be modified in order to disregard those original unary functions. Consider an arbitrary static variable ordering, that we assume lexicographic, *subproblem* P_i is the problem induced by variables $(x_i, \ldots, x_n)$. Formally, $P_i = ((x_i, \ldots, x_n), (D_i, \ldots, D_n), \{f \in \mathcal{F} \mid var(f) \subseteq \{x_i, \ldots, x_n\}\})$. Note that $P_1 = P$. Let $F_i(X_i^n)$ and $\mathcal{E}_i$ be the objective function and the efficient frontier of P_i, respectively. The essence of RDS is to replace the resolution of P by n successive resolutions on nested subproblems from P_n to P_1, where information from previous resolutions is used in the computation of new upper and lower bounds.

[2]For consistency with the next section, we assume that variables are selected in lexicographic order.

Multi-objective Russian Doll Search (MO-RDS) is described in Figure 2 (lines 12-17). It uses an *array of frontiers* rds. Frontier rds[i] plays the role of ubf during the resolution of subproblem P_i, so at the end of its resolution it will contain its efficient frontier $\mathcal{E}_i$. Thus, the efficient frontier of the original problem will be in rds[1]. The algorithm solves P_i in decreasing order of i (line 13). First, rds[i] is initialized with a valid upper bound frontier (line 14). Then, P_i is solved with a call to MO-BB (line 15), which computes a specific russian doll lbf. These two steps are further developed in the two following subsections.

Lower bound frontier

Consider an execution of MO-RDS. Let P_i be the subproblem that is being solved by MO-BB. Let x_j be the variable that is currently considered for assignment (namely, X_i^{j-1} and X_j^n are the set of assigned and unassigned variables, respectively). MO-BB uses the following lower bound frontier,

$$\texttt{lbf}_{rds} = \texttt{lbf}_s + \sum_{k=j}^{n} ic_k + \texttt{rds}[j]$$

which incorporates rds[j], the efficient frontier of P_j.

Theorem 1 *$\texttt{lbf}_{rds}$ is a valid lower bound frontier.*

Initial upper bound frontier

As noted in classical RDS, the resolution of P_{i+1} by MO-RDS contains useful information for the initialization of rds[i]. The idea is that any extension of an optimal assignment of P_{i+1} to variable x_i is likely to be near-optimal in P_i (because the two problems are very similar). Then, MO-RDS computes as initial upper bound frontier of P_i (Figure 2, line 14),

$$\texttt{ubf}_i = \| \bigcup_{\substack{a \in D_i \\ \vec{v} \in \texttt{rds}[i+1]}} F_i((x_i = a) \cdot A_{i+1}^n(\vec{v})) \|$$

where $A_{i+1}^n(\vec{v})$ is an optimal assignment with respect to P_{i+1} such that $F_{i+1}(A_{i+1}^n(\vec{v})) = \vec{v}$.

Theorem 2 *$\texttt{ubf}_i$ is a valid upper bound frontier.*

Property 1 *In a problem with a single objective function (i.e., $p = 1$), the algorithm MO-RDS is equivalent to RDS.*

Specialized MO-RDS

As shown in the mono-objective case, it may be convenient to ask russian doll to solve more subproblems in order to obtain more accurate lower and upper bound frontiers. Let P_{ia} be subproblem P_i in which the domain of x_i is restricted to value a. Formally, $P_{ia} = ((x_i, \ldots, x_n), (\{a\}, D_{i+1}, \ldots, D_n), \{f \in \mathcal{F} \mid var(f) \subseteq \{x_i, \ldots, x_n\}\})$. *Specialized* RDS (SRDS) (Meseguer & Sanchez 2001) solves each P_{ia}. Figure 2 (lines 18-26) describes *Specialized Multi-objective Russian Doll Search* (SMO-RDS). The algorithm solves P_{ia} for each variable i from n down to 1 and for each domain value $a \in D_i$ (lines 19-20). Each P_{ia} is optimally solved using MO-BB (line 22) and, at the end of the resolution, rds[i, a] is its efficient frontier. Note that the efficient frontier of P_i is $\texttt{rds}[i] = \| \bigcup_{a \in D_i} \texttt{rds}[i, a] \|$. Therefore, the efficient frontier of the overall problem is the union of non-dominated costs of P_{1a} for each $a \in D_1$ (line 26).

When solving P_{ia} and A_i^{j-1} (with $i < j$) being the current partial assignment, the lower bound frontier of MO-SRDS is,

$$\texttt{lbf}_{srds} = \texttt{lbf}_s + \sum_{k=j+1}^{n} ic_k + \Big\| \bigcup_{b \in D_j} \big(ic_{jb} + \texttt{rds}[j, b]\big) \Big\|$$

Theorem 3 *$\texttt{lbf}_{srds}$ is a valid lower bound frontier.*

The following theorem shows that the lower bound frontier of SMO-RDS is always tighter than the lower bound of MO-RDS.

Theorem 4 *$\texttt{lbf}_{rds} \leq \texttt{lbf}_{srds}$.*

Before solving P_{ia}, SMO-RDS initializes rds[i, a] with a valid upper bound frontier (line 21). The initialization follows the same idea as in MO-RDS. Each efficient solution of $P_{(i+1)b}$ is extended to variable x_i assigned to a. Formally,

$$\texttt{ubf}_{ia} = \| \bigcup_{\vec{v} \in \texttt{rds}[i+1]} \{F_i((x_i = a) \cdot A_{i+1}^n(\vec{v}))\} \|$$

Theorem 5 *$\texttt{ubf}_{ia}$ is a valid upper bound frontier.*

Experiments

We test the performance of MO-SRDS in the SPOT5 benchmark (Bensana *et al.* 1996). These instances model the scheduling of a satellite that orbits the earth while taking photographs requested by different customers. Since it is impossible to fulfil all the requests, the problem is to select the best subset of photographs that the satellite will actually take. The selected subset of photographs must satisfy a large number of imperative constraints and at the same time maximize the importance of selected photographs (which is modelled with unary cost functions). It is a mono-objective optimization problem.

SPOT5 instances can be divided into *single* and *multiple orbit*. In single orbit instances imperative constraints are binary and ternary. RDS and SRDS have proved to be the most efficient methods for this class of instances (Verfaillie, Lemaître, & Schiex 1996). Multiple orbit instances have an additional constraint imposed by the amount of memory required to store the different photographs and the on-board storage limit. This type of constraints are usually called *capacity* constraints and have the form $\sum_{i=1..n} c_i(x_i) < K$, where $c_i(x_i)$ is the memory consumption of decision x_i and K is the storage limit. Multiple-orbit instances are very challenging and remain unsolved (the only exception is instance 1502 where the capacity constraint is irrelevant because it can be trivially satisfied). In the following, we will focus on multiple orbit instances.

First, it is important to note that RDS and SRDS cannot solve these instances in reasonable time because the capacity constraint is ignored in the different subproblems P_i until

$i = 1$. Consequently, optimal solutions of the different subproblems P_i ($i > 1$) provide very low contributions to the lower bound of P: their ignorance of the capacity constraint allows them to take many space-consuming pictures.

A second observation is that these instances can be solved as if they were bi-objective, the memory consumption being the second objective. The solution of the original mono-objective instance is the efficient solution of the bi-objective version such that the value of the second objective does not surpass K, and the value of the first objective is minimum.

Solving these instances as bi-objective may seem a bad option, since we are only concerned with one point of the efficient frontier. However, there is a very important structural property: with the mono-objective point of view the interaction graph is connected (it is a clique, because the capacity constraint connects all the variables), but with the bi-objective point of view the interaction graph is made of several independent parts that can be solved independently. Note that any previous attempt to solve this instances has treated them as mono-objective and has failed.

In our experiments, we will refer to subproblems with the following notation: $X(i, j)$ denotes instance X restricted to the subset of consecutive variables $i \dots j$. When a subproblem coincides with a connected component of the overall problem (in the multi-objective sense), we will indicate it as $X(i, j)^*$. It is important to note that all the multiple orbit instances have been derived from the largest instance 1407 reducing either the number of variables or the number of constraints. As a result, it turns out that the same subproblem may have different names with our notation. When this is the case, we will name it as the subproblem coming from instance 1407.

In our comparison, we consider the following solving alternatives:

- ILOGSOLVER. A commercial solver that incorporates most state-of-the-art constraint programming techniques.
- MOBB$_{fdac}$. A branch-and-bound solver that enforces FDAC (Larrosa & Schiex 2003) independently in each objective function.
- MOBB$_{mombe}$. A branch-and-bound solver that uses *multi-objective mini-buckets* (Rollon & Larrosa 2006a) as a bounding evaluation function.
- MO-RDS and SMO-RDS.

We break the multiple orbit SPOT5 instances into subinstances and run the five algorithms on them with a time limit of one hour. Figure 3 reports the results. Subinstances which are connected components with less than 20 variables or with trivially satisfiable capacity constraints can be solved almost instantly by any method and we do not report them. The first column of the figure indicates the name of each instance. The second column tells the number of constraints. The following columns report cpu time in seconds required by each algorithm . Symbol "-" indicates that the algorithm cannot solve the subproblem within the time limit. As can be observed, ILOGSOLVER is the worst option and can only solve two of the instances. The explanation is that it has been conceived for constraint *satisfaction* problems

Instance	constr	Time (sec.)				
		ILOG	fdac	mombe	mo-rds	smo-rds
1506(0,150)	1440	-	-	-	-	2562
1506(151,228)	1107	-	-	-	424	2412
1506(229,317)	1349	-	-	-	90	62
1506(679,761)*	1243	-	-	-	1	1
1407(0,147)	1442	-	-	-	-	1625
1407(148,247)	1678		-	-	866	2545
1407(248,378)	3063	-	-	-	366	680
1407(379,409)*	220	-	2205	2	0	0
1407(413,429)*	87	18	2	0	0	0
1407(447,469)*	129	40	0	0	0	0
1407(494,553)*	129	-	-	-	267	2980
1407(580,696)	2299	-	-	-	-	1764
1407(700,761)	445	-	-	-	7	3
1407(762,878)*	2708	-	-	-	27	96

Figure 3: Experimental results on capacity constraints subproblems. Time limit 3600 seconds.

Instance	vars	constr.	lower bound
1506	940	15240	344556
1401	488	10963	441117
1403	665	13616	441271
1405	855	18258	441470
1407	1057	21786	441634

Figure 4: Lower bounds for SPOT5 instances. Time limit 3600 seconds.

and does not handle well the optimization nature of these instances. The second worst solver is MOBB$_{fdac}$. The lower bounding function of this solver has been proved very convenient in mono-objective optimization problems. However, when dealing with multi-objective problems it loses accuracy, because it treats each objective independently. Solvers MOBB$_{mombe}$, MO-RDS and SMO-RDS are the only ones whose lower bounding functions exploit the multi-objective nature of the problem. However, MOBB$_{mombe}$ performs poorly, because its lower bound is not very tight and does not prune well in this benchmark. MO-RDS and SMO-RDS are clearly the best alternatives for this benchmark. This is somehow expectable, since mono-objective RDS already proved its efficiency in the single orbit instances and our algorithms are a natural generalization. The efficiency of MO-RDS and SMO-RDS depends on the structure of each subinstance. When the lower bound of a given subproblem P_i is similar to the lower bound of each subproblem P_{ia}, MO-RDS outperforms SMO-RDS because the latter cannot take advantage of the specialized lower bound. However, when this is not the case, SMO-RDS computes a tighter lower bound and, as a result, SMO-RDS is more efficient than MO-RDS.

Figure 4 reports the lower bound that we obtain for each complete instance. The lower bound is computed as follows: for each complete instance we sum the efficient frontier of each subproblem. This is a lower bound frontier of the bi-objective version of the instance. Then, we discard from the frontier all the cost vectors that surpass the on-board storage limit. We then take the cost vector with the lowest value in

Instance	constr.	SMO-RDS (sec.)
1504(0,183)*	1329	1114
1504(184,206)*	112	0
1504(356,461)*	105	13418
1504(462,508)*	301	0

Figure 5: Experimental results on instance 1504.

the original objective function.

In our second experiment, we focus on the resolution of instance 1504. We solve to optimality each connected part of the problem using SMO-RDS with no time restriction. Figure 5 reports the cpu time for the most difficult subinstances. The most difficult piece, $1504(356, 461)$, is solved in less than 4 hours. As a consequence, we can sum the efficient frontier of each subproblem and extract the solution of the original problem: 161301. This instance is solved with SMO-RDS for the first time.

Related Work

Recently, some authors have considered multi-objective optimization inside of the constraint programming paradigm. For instance, (Rollon & Larrosa 2006a) extends an inference algorithm called *bucket elimination* to multi-objective problems. As could be expected, the algorithm suffers from a high space complexity and can only be used in problems with a small induced width. The alternative to inference is search. (Gavanelli 2002) introduces a basic branch and bound algorithm and proposes the use of quad-trees to store upper bound frontiers. His algorithm is similar to our MO-BB (Figure 2) using the simplest lower bound function. A more sophisticated lower bound function is *multiobjective mini-bucket elimination* (mo-mbe) (Rollon & Larrosa 2006a) which is an approximation of bucket elimination. The main contribution of our work is also the definition of a sophisticated lower bound, embeded in the mo-rds algorithm. One can easily observe that mo-rds is always stronger than mo-mbe. However, the time complexity of computing mo-rds is exponential, while the time complexity of computing mo-mbe is polynomial. It is important to note that both lower bounds has been proved to be useful for different classes of problems.

The idea of revealing and exploiting the *multi-objective nature* of problems that are not originally defined as multi-objective is not new. (Junker 2006) and (Rollon & Larrosa 2006b) apply this idea to pure constraint satisfaction problems (namely, without any objective function) in which some (at least two) of the constraints can be encoded as $F(X) < K$ and their nature makes us believe that it is difficult to satisfy all of them simultaneously. The main difference with respect our work is the lower bound used. (Junker 2006) uses the simplest lower bound and finds the efficient frontier in a given order. (Rollon & Larrosa 2006b) uses mo-mbe. Our approach uses mo-rds and smo-rds.

Conclusion

This paper has two contributions. The first one is algorithmic, since we extend RDS and SRDS from mono-objective to multi-objective optimization. The second contribution comes from our empirical work, where we show that our multi-objective RDS is an efficient alternative in an important class of problems. Moreover, using multiple orbit SPOT5 instances, we illustrate that sometimes it may be convenient to reformulate pure satisfaction or mono-objective optimization problems as multi-objective problems. This may be counterintuitive at first sight because multi-objective optimization is in general more difficult than mono-objective optimization. However, we show that the multi-objective perspective may bring to light desirable structural properties.

It is known that russian doll algorithms are very sensitive to heuristic decisions (Meseguer & Sanchez 2001). Our current implementation of SMO-RDS is far from being fully optimized with respect to available mono-objective versions[3]. Thus, we plan to accelerate our solver and improve our results on the multiple orbit SPOT5 instances. We expect to solve other open instances in the near future.

References

Apt, K. 2003. *Principles of Constraint Programming*. Cambridge: Cambridge University Press.

Bensana, E.; Verfaillie, G.; Agnse, J.; Bataille, N.; and Blumstein, D. 1996. Exact and inexact methods for the daily management of an earth observation satellite. In *In Proc. 4 th Intl. Symposium on Space Mission Operations and Ground Data Systems, Munich, Germany, 1996.*

Ehrgott, M., and Gandibleux, X. 2002. *Multiple Criteria Optimization. State of the Art. Annotated Bibliographic Surveys.* Kluwer Academic Publishers.

Freuder, E., and Wallace, R. 1992. Partial constraint satisfaction. *Artificial Intelligence* 58:21–70.

Gavanelli, M. 2002. An algorithm for multi-criteria optimization in csps. In *Proceedings of the 15th Eureopean Conference on Artificial Intelligence, ECAI'2002*, 136–140.

Junker, U. 2006. Preference-based inconsistency proving: When the failure of the best is sufficient. In *Proc. of ECAI'06*, 118–122.

Larrosa, J., and Schiex, T. 2003. In the quest of the best form of local consistency for weighted csp. In *Proc. of the 18^{th} IJCAI*, 239–244.

Meseguer, P., and Sanchez, M. 2001. Specializing russian doll search. In *CP'01*, 464–478.

Rollon, E., and Larrosa, J. 2006a. Bucket elimination for multiobjective optimization problems. *Journal of Heuristics* 12(4-5):307–328.

Rollon, E., and Larrosa, J. 2006b. Multi-objective propagation in constraint programming. In *Proc. of ECAI'06*, 128–132.

Rossi, F.; van Beek, P.; and Walsh, T. 2006. *Handbook of Constraint Programming*. Elsevier. chapter 9.

Verfaillie, G.; Lemaître, M.; and Schiex, T. 1996. Russian doll search. In *AAAI-96*, 181–187.

[3] ftp://ftp.cert.fr/pub/lemaitre/LVCSP