Finding Regions for Local Repair in Partial Constraint Satisfaction

Harald Meyer auf'm Hofe

German Research Center for Artificial Intelligence
Postfach 2080
D-67608 Kaiserslautern, Germany

Abstract. Yet, two classes of algorithms have been used in partial constraint satisfaction: local search methods and *branch&bound* search extended by the classical constraint-processing techniques like e.g. *forward checking* and *backmarking*. Both classes exhibit characteristic advantages and drawbacks. This article presents a novel approach for solving partial constraint satisfaction problems exhaustively that combines advantages of local search and extended *branch&bound* algorithms. This method relies on repair based search and a generic method for an exhaustive enumeration of repair steps.

1 Introduction

Algorithms for solving *constraint satisfaction problems (CSP)* have been successfully applied to several fields including scheduling, design, and planning. Extending the standard CSP by a representation of constraint importance led to the class of *partial constraint satisfaction problems (PCSP)* [2] which provides new opportunities for solving several problems of combinatorial optimization more efficiently. A $PCSP$ is the task of labeling each variable of a given variable set with a value of a certain domain. Constraints of an explicitly represented importance state restrictions on combinations of some variables' labels. The solution of a $PCSP$ is a labeling complying with as important constraints as possible.

Yet, two classes of algorithms have been applied to partial constraint satisfaction: local search methods [9] and systematic *branch&bound* search extended by classical constraint-processing techniques like *forward checking* and *backmarking* [2, 4, 7]. Both paradigms exhibit characteristic advantages and drawbacks. Theoretically, the *branch&bound* algorithm is guaranteed to terminate with an optimal solution. However, tree search algorithms retract early decisions only after searching large portions of the search space exhaustively. As a consequence, minor differences in the constraint problem can result in a completely different run time behavior — especially if the number of variables is larger. In contrast, local search procedures try to improve a labeling without respecting a certain systematic in search. Thus, they are applicable to dynamic constraint problems, where variables and constraints are added resp. removed. However, the computed result is of a questionable quality. Proving the optimality of a result is

not possible by use of local search algorithms. Local minima are processed by heuristics which either make use of noise strategies (*mincon-walk* and *mincon-retry* [12]), exploit information on the search history (e.g., tabu search [3]), or change more than one variable at each step of repair (e.g., EFLOP-heuristics [13] or the *lc*–algorithm [11]).

This paper presents a novel approach for solving partial constraint satisfaction problems that combines the advantages of local search and tree search algorithms. This approach is based on iterative improvement of an initially chosen labeling repeating the following steps: A region (a set of variables) in the constraint problem is chosen by an exhaustive enumeration strategy. Then, an extended *branch&bound* is used to optimize the labels in this region. This method organizes the search space flexibly like local search algorithms, i.e. the algorithm is able to change any assignment at any time if this promises to lead to an improvement in the quality of the labeling. Generally, iterative improvement algorithms have to be tailored to the application. Exhaustive enumeration of regions for local repair turns iterative improvement into a generic algorithm and provides the ability to prove optimality of a solution.

The paper is organized as follows: After providing some basic definitions on partial constraint satisfaction, the second section describes iterative improvement algorithms. A section on exhaustive enumeration of iterative repair steps follows, which starts with a small theory on enumerating improvement steps and ends with an enumeration algorithm. Then, first empirical results are presented in order to prove the relevance of the approach. A concluding section sums up the results.

2 Partial Constraint Satisfaction

In order to clarify notations, which are especially required in section 4, some basic definitions are given here.

Definition 1. *A CSP is a tuple* (V, D, C) *where* V *is a set of variables, the domain* D *is a set of values which can be assigned to the variables, and* C *is a set of constraints, where each* $c \in C$ *is defined by:* $V(c) \subseteq V$ *is a set of variables which are* directly affected *by* c*. The extension of* c, $\mathrm{ext}(c)$*, is a set of labelings of all variables in* $V(c)$ *with values of* D *which comply with* c*.*

$$l\downarrow_{V'} \;=\; \{v_i \;\leftarrow\; d_i \in l | v_i \in V\}$$

denotes the selection of labels which concern the variables in V'*. A labeling* l *complies with a constraint* c *iff* $l\downarrow_{V(c)} \;\in\; \mathrm{ext}(c)$*.*

$$\bar{C}(l) \;=\; \{c \in C \mid l\downarrow_{V(c)} \;\notin\; \mathrm{ext}(c)\}$$

denotes the set of constraints being violated by l*. A labeling* l *of all variables in* V *is a* solution *of* $CSP = \{V, D, C\}$ *iff*

$$l \in \mathrm{ext}(C) \Longleftrightarrow \forall c \in C : l\downarrow_{V(c)} \in \mathrm{ext}(c)$$

$$\iff \bar{C}(l) = \{\}$$
$$\iff l \in \bowtie_{c \in C} \mathrm{ext}(c).$$

Hence, a solution of a CSP is a labeling of all variables which complies with all constraints.

In partial constraint satisfaction[1], the relative importance of constraints is given by a partial ordering among constraint sets where $C' \succ C''$ means intuitively: *The constraints in C' are more important than the constraints in C''*. The solution of a $PCSP$ satisfies as important constraints as possible.

Definition 2. *A $PCSP = (V, D, C, \succ)$ extends a $CSP = (V, D, C)$ by a preference ordering $\succ$, which is a partial ordering among subsets of C. A labeling l of all variables in V is a solution of a $PCSP$ iff there is no labeling $l' \neq l$ with $\bar{C}(l) \succ \bar{C}(l')$.*

This notion of PCSPs forms the domain of the *enumerating global revisions* method. The well known algorithms for solving $PCSP$s like the *branch&bound* [2] as well as *mincon* [9] and *mincon-walk* [12] can be adopted easily to these definitions [6].

3 Iterative Improvement

Fig. 1 presents a scheme for searching by *iterative improvement* which forms the basis for our novel approach. A labeling l of all variables is modified iteratively repeating the improvement step within the rows 2 to 5. This method may be considered as a generalization of the *mincon* algorithm [9] where, in contrast to *mincon*, local minima are passed conducting more than one change in the current labeling l within a single step of repair.

Therefore, a procedure `choose-bad-region` is used to determine a set of variables whose labels will be changed in the following improvement step (row 2). The constraint graph of the problem, the domains of the variables[2], the current labeling, and the set of violated constraints are useful parameters of this procedure. In practical applications of such algorithms, e.g., in our commercial nurse scheduling system [8, 5], `choose-bad-region` is tailored to the current application. In contrast, this paper introduces a generic algorithm for this procedure.

Then, the improvement step is conducted by a variant of the *branch&bound* in row 4 that changes the labels of the variables V' in labeling l to improve the overall quality of l. The current set of violated constraints is used as initial bound because the new labeling l (after the improvement step) is not allowed to be worse than the old one (before the improvement step). Constraint propagation like *forward checking* can be employed in order to increase performance of the improvement step.

[1] In the sense of this paper.

[2] These domains can be the result of an initial arc-consistent filtering that has been conducted before search.

iterative-improvement(V, D, C, $\succ$)

1. Compute an initial labeling l of all variables in V;
2. $V' =$ choose-bad-region$(V, C, \succ, l, \bar{C}(l))$;
3. if $V' = \emptyset$ then go to 7;
4. unassign the variables in V', propagate all constraints c with $V(c) \cap V' \neq \{\}$ and run *branch&bound* on the variables in V' with $\bar{C}(l)$ as initial bound in order to compute a new labeling l;
5. add temporary hard constraint

$$\bigvee_{v \in V \setminus V'} v \neq l\downarrow_v$$

6. go to 2;
7. remove temporary constraints. return l as result.

Fig. 1. Searching by iterative improvement.

Branch&bound searches all possible labelings of V' exhaustively. Consequently, further improvement steps have to consider a change in $V \setminus V'$ in order to prevent the search algorithm from visiting labelings more than once. This condition is enforced in row 5 by an additional temporary compulsory constraint.

As all local search algorithms, *iterative improvement* is applicable to *dynamic constraint satisfaction problems (DCSP)*, where constraints and variables are added to or removed from the current constraint problem between two calls of the search procedure. Applications of dynamic constraint satisfaction usually require consecutive search processes on related problems to result in similar solutions which share as many labels is possible. This notion of stability is provided for free by local search algorithms like *iterative improvement* that accept only an improvement of current labeling as valid repair steps.

This algorithm — enhanced by an heuristic `choose-bad-region` procedure — does a pretty good job, e.g. on nurse scheduling problems, where *mincon-walk* as well as EFLOP-like heuristics failed at producing solutions of sufficient quality [8, 5]. Nevertheless, applying heuristics to the selection of *bad regions* in partial solutions of a PCSP exhibits some remarkable drawbacks. Firstly, the applied heuristics consider only a few constraints when choosing variables for reselection. Secondly, the heuristics heavily rely on assertions on the constraint model which makes the adoption of the system to slightly different applications a non-trivial problem. These deficiencies motivated to investigate opportunities for deriving similar heuristics directly from the constraint model.

4 Exhaustive Enumeration of Improvement Steps

A naive method for avoiding the application of search heuristics in *iterative improvement* would enumerate *all possible* regions by the `choose-bad-region` procedure called in `iterative-improvement` (cf. Fig. 1). When called by the

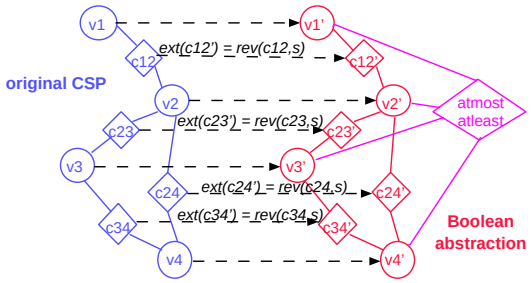

Fig. 2. Local revision sets form constraints on possible revisions of the whole problem with reference to the current labeling S.

overall search algorithm, this procedure first returns all sets of only one variable. If all of these variable sets have been enumerated, then all sets of two variables are returned and so on. Finally, after trying to improve $V' = V$, one knows that a global solution has been computed. However, this method neglects all information that can be retrieved from the constraint graph and the deficiencies of the current solution.

Our idea about a generic procedure for finding bad regions in partial solutions of a PCSP is to constrain this naive enumeration of regions for solution improvement by the available information. In the following, promising regions for solution improvement are called to form global revision sets. The problem of finding global revisions is formulated as a Boolean PCSP whose constraint graph is very similar to the original problem (cf. Fig. 2). If a solution of the Boolean problem assigns a 1 to a variable v_i', the corresponding variable v_i in the original problem is considered to be a part of a promising region for improvement. Each of the constraints c_{ij}' in this Boolean problem represents a necessary condition on promising regions that only depends on *one* constraint c_{ij} in the original problem and a partial solution S. In the following, the regions complying with the condition concerning a single constraint are called to define a set of local revisions. The constraints *atmost* and *atleast* are well known from scheduling systems and count here the occurrences of 1 in the solution of the binary problem. These constraints can be used to control the size of the regions. Hence, it is possible to return small regions first in order to conduct cheap improvement steps first.

4.1 Local Revision Sets

Firstly, the relation of the original problem to the abstract problem of finding regions for local repair according to Fig. 2 needs to be defined. In the following, v' always denotes the variable in the abstract Boolean problem that corresponds to variable v in the original problem. Analogously, c' represents the constraint in the abstract problem referring to constraint c in the original problem. The whole

set of variables in the abstract problem is written as V_{rev}, the set of constraints as C_{rev}.

Definition 3. *Let l_1 and l_2 be labelings of all variables in V. Then* $\mathrm{diff}(l_1,\ l_2)$ *returns a labeling of V_{rev} such that*

$$\forall v' \in V_{rev} : \mathrm{diff}(l_1,\ l_2)\downarrow_{v'} = \begin{cases} 0 & \textit{iff } l_1\downarrow_v = l_2\downarrow_v \\ 1 & \textit{otherwise.} \end{cases}$$

Let l be a labeling of the variables in V and c be a constraint in the original problem. Then, the smallest local revision set of l respecting c is defined as follows:

$$\mathrm{rev}(c,\ l) := \bigcup_{l' \in \mathrm{ext}(c)} \{\mathrm{diff}(l\downarrow_{V(c)},\ l')\}.$$

All extensions comprising $\mathrm{rev}(c,\ l)$ *are called local revision sets of l respecting c.*

Local revision sets cover all differences of a current labeling l from all labelings complying with constraint c. Although defined extensionally here, intensional definitions for common constraints, which are typically defined by propagation methods in a constraint library, are possible.

4.2 Global Revision Sets

Global revision sets consider all opportunities to satisfy a set of constraints instead of single constraints.

Definition 4. *The* smallest *revision set of a partial solution l respecting constraint set C' is defined as*

$$\mathrm{rev}(C',\ l) := \bigcup_{l' \in \mathrm{ext}(C')} \{\mathrm{diff}(l,\ l')\}.$$

All extensions comprising $\mathrm{rev}(C',\ l)$ *are called revision sets of l respecting C'. Revision sets respecting all constraints in C are called global revision sets.*

Lemma 1.

$$\bowtie_{c \in C'} \mathrm{rev}(c,\ l) \supseteq \mathrm{rev}(C',\ l).$$

Obviously, the join of local revisions above denotes exactly the set of all solutions to the abstract problem in Fig. 2. Hence, the lemma claims that the solutions of this abstract constraint problem form a global revision set of labeling l. However, this global revision set is generally not minimal.

Proof. One can prove by a few equations that for any $l' \in \mathrm{ext}(C')$ the difference from the current label $\mathrm{diff}(l',\ l)$ is in $\bowtie_{c \in C'} \mathrm{rev}(c,\ l)$ presupposing that all variables in l are directly affected by a constraint in C':

$$\begin{aligned} & \bowtie_{c \in C'} \mathrm{rev}(c,\ l) \\ = {} & \bowtie_{c \in C'} (\{\mathrm{diff}(l'\downarrow_{V(c)},\ l\downarrow_{V(c)})\} \cup \mathrm{rev}(c,\ l)) \\ = {} & \{\mathrm{diff}(l',\ l)\} \cup \bowtie_{c \in C'} \mathrm{rev}(c,\ l). \end{aligned}$$

The first equation follows from the definition of local revision sets. The second presupposes that all variables in V are directly affected by the constraints in C' and

$$\mathrm{diff}(l' \downarrow_{V(c)},\ l \downarrow_{V(c)}) \in \mathrm{rev}(c,\ l).$$

□

Proposition 1.

$$\bowtie_{c \in C} \mathrm{rev}(c,\ l) \supseteq \mathrm{rev}(C',\ l)$$

holds true for all most important constraint sets $C' \subseteq C$ in a PCSP which can be satisfied.

Proposition 1 claims in other words that each globally optimal solution can be reached changing the regions as indicated by the global revision sets.

Proof. The inclusion $\bowtie_{c \in C}\ \mathrm{rev}(c,\ l) \supseteq \mathrm{rev}(C,\ l)$ follows from the lemma.
$C \supseteq C' \implies \mathrm{rev}(C,\ l) \supseteq \mathrm{rev}(C',\ l)$ is implied by definition 4.
□

4.3 Searching Global Revisions

The basic idea for applying these results is to search the original problem by algorithm `iterative-improvement` (according to Fig. 1) controlled by an exhaustive search of the abstract problem that enumerates global revisions. The resulting hybrid algorithm still is an anytime-algorithm because partial solutions are available all the time. Additionally, proving optimality is possible due to the results of the previous section. If all global revisions have been searched without improving the current partial solution l, then l is optimal. Fig. 3 describes the idea of an enumeration algorithm for partial constraint satisfaction problems. $PCSP_{rev}$ holds the constraint problem for finding global revisions. As mentioned above, *atmost* and *atleast* constraints are used to enumerate global revisions according to their size in order to conduct cheap improvement steps first in algorithm `iterative-improvement`. The variable n, occurring in the rows 1 and 5, controls the number of assignments to be retracted. In row 1, $PCSP_{rev}$ is built up again after improving l because the constraints in C_{rev} typically depend on l. Variable δ_{rev}, which is set in row 2, serves as an initial bound in the following call of the *branch&bound* procedure. Only revisions promising to improve l are returned by the enumeration in row 4. If the *branch&bound* fails to find a new partial solution of $PCSP_{rev}$ better than δ_{rev} then, occasionally, larger revisions are required (row 5). If no larger revisions are available then l is optimal. Search in `iterative-improvement` terminates because `choose-bad-region` returns an empty variable set.

Enumerating regions ordered by their size is only one example for an enumeration strategy. There are many alternative strategies conceivable which for instance may try to find revisions first that promise to repair the most important currently violated constraints [6]. Additionally, the degree of prospective constraint processing, which is used searching the abstract problem by

choose-bad-region(V, C, $\succ$, 1, δ)

1. if $PCSP_{rev}$ has no value (this function is called for the first time) or δ has been improved by the last improvement step in iterative-improvement then begin
   (a) $PCSP_{rev} := (V_{rev}, \{0,1\}, C_{rev}, \succ')$ with $C_{rev} = \bigcup_{c \in C} \mathrm{rev}(c,\ l)$ and $C_1 \ \succ' \ C_2 \iff \{c \mid c' \in C_1\} \succ \{c \mid c' \in C_2\}$;
   (b) $n := 1$;
   (c) add the following constraint to $PCSP_{rev}$: *atleast* and *atmost* n occurrences of 1
   end;
2. $\delta_{rev} := \bigcup_{c \in \delta} \mathrm{rev}(c,\ l)$;
3. call *branch&bound* with δ_{rev} as bound on $PCSP_{rev}$ for the *next* partial solution better than δ_{rev};
4. if a partial solution l has been found return $V' = \{v \mid l \downarrow_{v'} = 1\}$ and exit;
5. if no partial solution is available and $n \leq \mid V \mid$ then do begin
   (a) $n := n + 1$;
   (b) reset $PCSP_{rev}$ and remove *atleast* and *atmost* constraints;
   (c) add the following constraints to $PCSP_{rev}$: *atleast* and *atmost* n occurrences of 1;
   (d) goto row 3
   end;
6. l is optimal. Hence, return $V' = \{\}$ and exit.

Fig. 3. Enumerating global revisions (egr).

branch&bound, is obviously relevant for the performance of the enumeration. If the *branch&bound* looks ahead deeply into the search space, it will find that global revisions first which promise to repair strong deficiencies. However, the overhead for searching global revisions is increased.

5 Experiences

Enumeration of global revisions (egr) comprises a whole family of algorithms where instances differ in the used enumeration strategy and the applied constraint techniques. The major drawback of this method is the overhead which is caused by the effort of searching the abstract constraint problem. Experiments on unstructured randomly generated constraint problems are presented here to give a first answer to the question: *Is the effort of enumerating global revisions acceptable?*

A comparative analysis of *egr* has to consider the two major tasks in partial constraint satisfaction: finding optimal solutions (exhaustive search) and finding as good solutions as possible within a limited amount of time (approximative optimization).

Table 1. Empirical comparison of *bb+FC* and *enumerating global revisions (egr)* searching exhaustively. Each row presents results of 10 runs on random problems for each algorithm.

no.	den.	sat.	algo.	time/s			checks/10^6			assignments/10^3		
				⌀	min.	max	⌀	min.	max	⌀	min.	max
30 variables, domains of 10 values, 6 hierarchy levels												
1	0.44	0.7	egr+FC	5282	1744	24083	111	34	535	119	38	564
			egr+MACall	*4019*	704	11777	146	26	508	153	30	535
			bb+FC	4348	28	12785	139	62	336	133	62	323
2	0.44	0.5	*egr+FC*	*7002*	396	15926	168	82	371	228	13	489
			egr+MACall	7022	854	21397	174	20	529	232	28	654
			bb+FC	10537	2678	31590	284	76	857	385	100	1182
3	0.22	0.5	egr+FC	824	92	3169	14.8	1.7	52.8	44	5	161
			egr+MACall	*755*	208	2522	15.7	4.2	57.9	44	12	156
			bb+FC	1047	547	2294	61.4	16.7	304.6	145	46	635

In the following, variants of *egr* are compared to standard algorithms due to experiments on *hierarchical constraint satisfaction problems (HCSP)*. An HCSP is simply a special kind of a PCSP where constraints are grouped into the hierarchy levels C_0 (comprising compulsory constraints) to C_n [1, 5, 6]. Additionally, $\omega(c)$ assigns a real number as a constraint weight to each constraint c. To decide whether $C' \succ C''$ holds for the constraint sets C' and C'', one considers the weight sum of the constraints in $C' \cap C_1$ and $C'' \cap C_1$ first. If C' has got more important constraints in hierarchy level 1 then $C' \succ C''$ holds. If C' comprises less important constraints in level 1 then the opposite holds. Otherwise, the next hierarchy level is concerned. Hierarchy levels may be considered as a categorical degree of a constraint's importance whereas the weight is a gradual measure of importance. HCSPs turned out to be useful for representing real world problems [8, 5] and, additionally, enable the use of certain *looking ahead* techniques and *dynamic variable orderings (DVO)* [6, 7]. As usual, constraint problems are classified according to the number of variables, constraint density (den.) and the constraint's satisfiability (sat.).

Exhaustive search: Table 1 presents empirical results on two variants of *egr* in comparison with the *branch&bound* algorithm *bb+FC*. Algorithm *bb+FC* uses *forward checking* equivalently to the P-FC3 procedure in [2] combined with a dynamic variable ordering heuristic, which is especially appropriate to constraint hierarchies [6, 7]. Forward checking results are used to assign best values first as value assignment strategy. Both *egr* procedures, *egr+FC* and *egr+MACall*, deploy *bb+FC* within the improvement step. Whereas, *egr+FC* also uses *bb+FC* searching the abstract problem, *egr+MACall* additionally applies a MAX-MIN-algorithm [10] after each assignment in the abstract problem which labels each value with the most important hierarchy level that cannot be satisfied completely assigning that value. The results of constraint propagation are used to inform the

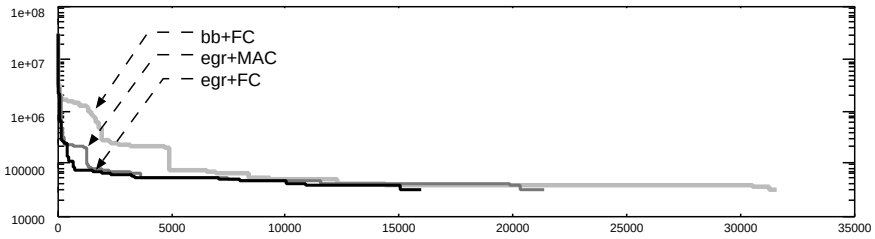

Fig. 4. Performance curve to experiment 2. Improvement in solution quality over time.

value selection strategy and the dynamic variable ordering heuristic (for details refer to [6]). This method has been included into the experiments to assess the effect of a larger amount of constraint propagation in the abstract problem which presumably increases the merit of the returned regions but causes additional costs in searching the abstraction.

Additional experiments, which are not listed in Fig. 1, turned out that *egr* algorithms are usually not recommended when searching smaller optimization problems comprising 20 variables. On problems of about 30 variables, *egr* algorithms are competitive to *bb+FC*. Apparently, spending more effort on looking ahead while searching for promising global revisions like in *egr+MACall* pays for itself when searching for optimal solutions of problems of this size.

Fig. 4 shows the major advantage that *egr* methods had in all experiments we have conducted. Solutions are improved more quickly. Fig. 4 displays the improvement in solution quality (y-axis) over time (x-axis). Therefore, the constraint hierarchy is transformed into an equivalent system of weighted constraints. A point (x, y) in a curve of Fig. 4 reports that constraints of weight y have been violated by the best yet found labeling at time x. Very similar curves result from the other experiments.

Approximative optimization: Approximative optimization of unstructured randomly generated problems is the domain of *mincon-retry* and *mincon-walk*. Fig. 5 presents the performance of *egr+FC* in comparison with *bb+FC*, *mincon-retry*, and *mincon-walk*. Forward checking results have been used to improve the initial labeling of local search algorithms in order to provide the same starting point as used in the *egr* algorithms.

Diagrams a) and b) in Fig. 5 show results on problems comprising 40 variables with a time limit of 5 minutes for each experiment. The diagrams c) and d) report the performance on larger problems comprising 100 variables with a time limit of 15 minutes for each experiment[3].

Obviously, *egr* behaves similar to *mincon-retry* and *mincon-walk*. The best yet found labeling is improved continuously without necessity to respect a certain systematic in search. Although searching two constraint problems instead of one, *egr* is more or less as fast as the standard algorithms in local search. However,

[3] A more detailed report on experiments of this kind can be found in [6].

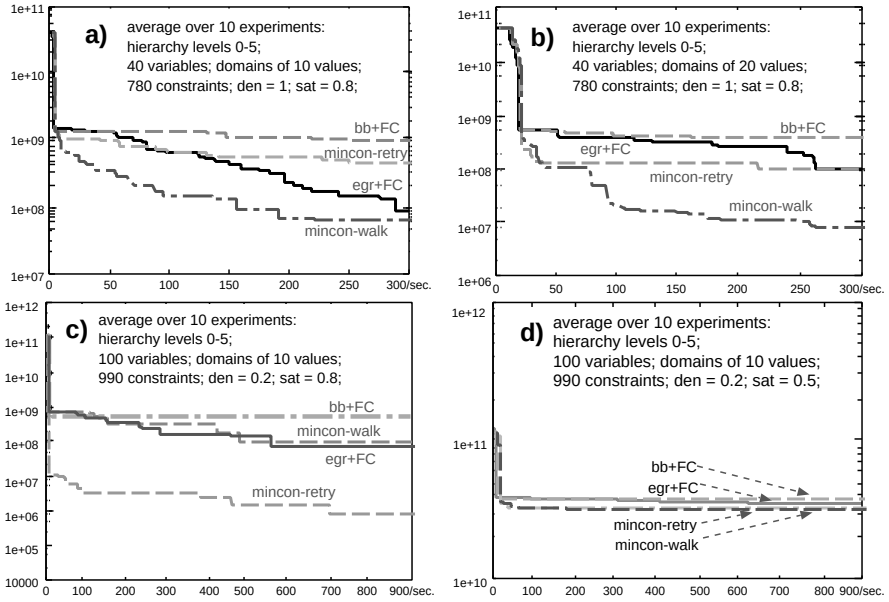

Fig. 5. Comparison of the decrease in the weight of violated constraints over time achieved by distinguished algorithms. The constraint problems vary in their size, density and the satisfiability of constraints.

on unstructured problems *mincon-retry* and *mincon-walk* perform often better than *egr*.

6 Summary

This paper introduced a novel method for turning repair-based search of partial constraint satisfaction problems into an exhaustive search procedure obtaining the advantages of local search: flexible improvement of the best yet found labeling and applicability to dynamic constraint satisfaction. This method, named *enumeration of global revisions (egr)*, relies on an enumeration of promising repair steps which is informed only by information which follows straightforward from the given constraint problem. First empirical evaluation on unstructured randomly generated constraint problems proves a general applicability of this method to both, exhaustive search of larger problems comprising about 30 variables and approximative optimization of even larger problems.

Further research aims at verifying this promise by practical application of *egr* to real world problems. Therefore, the major drawback of this method has to be attacked: the large implementation effort. For each constraint, which is used in the target application, a Boolean abstraction has to be implemented, which restricts the enumeration of revisions. As soon as constraint libraries of

the required expressiveness exist, *egr* will have to prove its applicability in time tabling and knowledge-based configuration.

References

1. Alan Borning, Bjorn Freeman-Benson, and Molly Wilson. Constraint hierarchies. *Lisp and Symbolic Computation*, 5:233–270, 1992.
2. Eugene C. Freuder and Rick J. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70, 1992.
3. Philippe Galinier and Jin-Kao Hao. Tabu search for maximal constraint satisfaction problems. In *CP–97: Proceedings of the Third International Conference on Principles and Practice of Constraint Processing*, volume 1330 of *LNCS*. Springer Verlag, pages 196–208, 1997.
4. Manfred Meyer. *Finite Domain Constraints: Declarativity meets Efficiency, Theory meets Application*. Dissertation, Universität Kaiserslautern, Infix Verlag, Sankt Augustin, 1994.
5. Harald Meyer auf'm Hofe. ConPlan/ SIEDAplan: Personnel assignment as a problem of hierarchical constraint satisfaction. In *PACT-97: Proceedings of the Third International Conference on the Practical Application of Constraint Technology*, pages 257–272, London, UK, April 1997. Practical Application Company, Ltd.
6. Harald Meyer auf'm Hofe. Finding regions for local repair in hierarchical constraint satisfaction. Research Report RR-97-05, Deutsches Forschungszentrum für Künstliche Intelligenz, December 1997.
7. Harald Meyer auf'm Hofe and Andreas Abecker. Zur Verarbeitung "weicher" Constraints. *KI – Künstliche Intelligenz*, (4):31–36, 1997.
8. Harald Meyer auf'm Hofe and Enno Tolzmann. ConPlan/ SIEDAplan: Personaleinsatzplanung als Constraintproblem. *KI – Künstliche Intelligenz*, (1):37–40, 1997.
9. Steven Minton, Mark Johnston, Andrew Philips, and Philip Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction problem and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.
10. Paul Snow and Eugene C. Freuder. Improved relaxation and search methods for approximate constraint satisfaction with a maximin criterion. In *Proc. of the 8^{th} biennal conf. of the canadian society for comput. studies of intelligence*, pages 227–230, May 1990.
11. G. Verfaillie and T. Schiex. Solution reuse in dynamic constraint satisfaction problems. In *AAAI–94: Proceedings of the 12th national conf. on AI*, pages 307–312, Seattle, WA, August 1994. AAAI Press/ The MIT Press.
12. Richard J. Wallace. Analysis of heuristic methods for partial constraint satisfaction problems. In *CP–96: Proceedings of the Second International Conference on Principles and Practice of Constraint Processing*, volume 1118 of *LNCS*. Springer Verlag, pages 482–496, 1996.
13. N. Yugami, Y. Ohta, and H. Hara. Improving repair-based constraint satisfaction methods by value propagation. In *AAAI-94: Proceedings of the 12th National Conference on Artificial Intelligence*, pages 344–349, 1994.