

# Recombination Operators for Satisfiability Problems

Lardeux Frédéric\*

Saubion Frédéric\*

Hao Jin-Kao\*

\*LERIA, University of Angers  
2 Bd Lavoisier, 49045 Angers Cedex 01  
{lardeux,saubion,hao}@info.univ-angers.fr

## 1 Introduction

The satisfiability problem (SAT) [6] consists in finding a truth assignment that satisfies a well-formed Boolean expression. An instance of the SAT problem is defined by a set of boolean variables (also called atoms)  $\mathcal{X} = \{x_1, \dots, x_n\}$  and a boolean formula  $\phi: \{0, 1\}^n \rightarrow \{0, 1\}$ . A literal is a variable or its negation. A (truth) assignment is a function  $v: \mathcal{X} \rightarrow \{0, 1\}$ . The formula is said to be satisfiable if there exists an assignment satisfying  $\phi$  and unsatisfiable otherwise. The formula  $\phi$  is in conjunctive normal form (CNF) if it is a conjunction of clauses where a clause is a disjunction of literals. In this paper,  $\phi$  is supposed to be in CNF. SAT is originally stated as a decision problem but we are also interesting here in the MAX-SAT problem which consists in finding an assignment which satisfies the maximum number of clauses in  $\phi$ .

SAT has many practical applications (VLSI test and verification, consistency maintenance, fault diagnosis, planning ...). During the last decade, several improved solution algorithms have been developed and important progress has been achieved. These algorithms can be divided into two main classes:

**Complete algorithms:** Designed to solve the initial decision problem, the most powerful complete algorithms are based on the Davis-Putnam-Loveland procedure [2]. They differ essentially by the underlying heuristic used for the branching rule [11, 17]. Specific techniques such as symmetry-breaking, backbone detecting or equivalence elimination are also used to reinforce these algorithms [1, 4, 10].

**Incomplete algorithms:** They are mainly based on local search [13, 15] and evolutionary algorithms [3, 5, 7]. Walksat [13] and UnitWalk [9] are famous examples of incomplete algorithms. Though incomplete algorithms are little helpful for unsatisfiable instances, they represent the unique approach for finding models of very large instances.

In this paper, we focus on the design of evolutionary based algorithms. An important process in evolutionary computation is the possibility of recombining interesting configurations of the problem. We present here a study of possible specific recombination operators that can be used in evolutionary algorithms for solving SAT problems.

Kyoto, Japan, August 25–28, 2003

## 2 Hybrid Evolutionary Algorithm

In this section, we define the main lines of a hybrid evolutionary algorithm obtained by combining a recombination stage with a local search improvement. Given an initial population, the first step consists in selecting its best elements (i.e. assignments) w.r.t. a fitness function *eval*. Then, recombinations are performed on this selected population. Each child is individually improved using a local search process and inserted under certain conditions in the population. The general algorithm is described in figure 1.

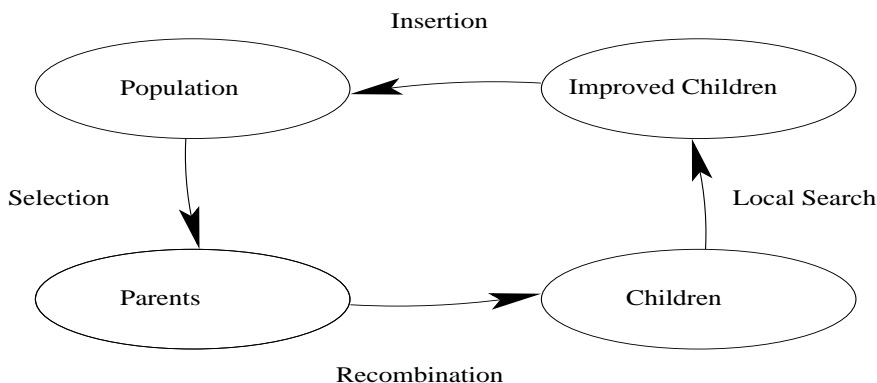


Figure 1: Algorithm Scheme

This algorithm can be adjusted by changing the selection function, the recombination or the local search method but also by modifying the insertion conditions. The whole algorithm relies on the management of a population of configurations representing potential solutions. Therefore, we present now the basic materials for this population and a brief description of the insertion and local search stage.

**Representation** The most obvious way to represent an individual for a SAT instance with  $n$  variables is a string of  $n$  bits where each variable is associated to one bit. Other representation schemes are discussed in [7]. Therefore the search space is the set  $\mathcal{S} = \{0, 1\}^n$  (i.e. all the possible strings of  $n$  bits) and an individual  $X$  obviously corresponds to an assignment.  $X[i]$  denotes the truth value of the  $i^{\text{th}}$  atom and  $X[i \leftarrow \alpha]$  denotes an individual  $X$  where the  $i^{\text{th}}$  atom has been set to the value  $\alpha$ . Given an individual  $X$  and a clause  $c$ , we use  $\text{sat}(X, c)$  to denote the fact that the assignment associated to  $X$  satisfies the clause  $c$ .

**Fitness Function** Given a formula  $\phi$  and an individual  $X$ , the fitness of  $X$  is defined to be the number of clauses which are not satisfied by  $X$ :

$$\text{eval}: \mathcal{S} \rightarrow \mathbb{N}, \text{eval}(X) = \text{card}(\{c \mid \neg \text{sat}(X, c) \wedge c \in \phi\})$$

where *card* denotes as usual the cardinality of a set. The smallest value of this function equals 0 and an individual having this fitness value corresponds to a solution.

**Selection Process** The selection operator is a function which takes as input a given population and extracts some individuals which will serve as parents for the recombination stage. To insure an efficient search, it is necessary to keep some diversity in the population. Indeed,

Kyoto, Japan, August 25–28, 2003

if the selected parents are too similar, some region of the search space  $\mathcal{S}$  will not be explored.

**Local Search** The local search process improves a configuration by a sequence of small changes in order to reach a local optimum w.r.t. the evaluation function and a good neighborhood function [16].

**Insertion** An improved child can be inserted w.r.t. different conditions (e.g. if it is better than any individual of the population). These conditions are important w.r.t. the average equality and the diversity of the population.

We now focus on the recombination stage.

### 3 Recombination Operators

A recombination operator (also called crossover) has to take into account as much as possible the semantics of the individuals in order to control the general search process and to obtain an efficient algorithm. In the SAT problem, the variables are the atoms and a constraint structure is induced by the clauses. Therefore, while the representation focuses on the atoms, an efficient crossover should take into account the whole constraint structure.

We first define a function  $flip: \{0, 1\} \rightarrow \{0, 1\}$  such that  $flip(x) = 1 - x$ . Then, we define a function  $imp: \mathcal{S} \times \mathcal{N} \rightarrow \mathcal{N}$  such that  $imp(X|i) = card(\{c \mid sat(X[i \leftarrow flip(X|i)], c) \wedge \neg sat(X, c)\}) - card(\{c \mid \neg sat(X[i \leftarrow flip(X|i)], c) \wedge sat(X, c)\})$ . This function computes the improvement obtained by the flip of the  $i^{th}$  component of  $X$  and was previously introduced in GSAT and Walksat [14, 13]. It corresponds to the gain of the solution according to the function  $eval$  (i.e. the number of false clauses which become true by flipping the atom  $i$  minus the number of satisfied clauses which become false). Remark that if this number is negative then the number of false clauses would increase if the flip is performed.

Each operator is a function  $cross: \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$  (i.e. two parents produce one child). We now describe how we construct the child  $C = cross(X, Y)$ .

#### Definition 1 Uniform Crossover

*Each variable takes the value of  $X$  or  $Y$  with equal probability.*

#### Definition 2 CCTM (Corrective Clause and Truth Maintenance) crossover

*For each clause  $c$  such that  $\neg sat(X, c) \wedge \neg sat(Y, c)$  and for all positions  $i$  such that the variable  $x_i$  appears in  $c$ , we compute  $\sigma = imp(X|i) + imp(Y|i)$  and we set  $C|k = flip(X|k)$  where  $k$  is the position such that  $\sigma$  is maximum. For all the clauses  $c$  such that  $sat(X, c) \wedge sat(Y, c)$  and for all positions  $i$  such that the variable  $x_i$  appears in  $c$ , we compute  $\sigma = imp(X|i) + imp(Y|i)$  and we set  $C|k = X|k$  where  $k$  is the position such that  $\sigma$  is minimum. All the variables with no value take the value of  $X$  or  $Y$  with equal probability.*

#### Definition 3 CC (Corrective Clause) crossover

*For each clause  $c$  such that  $\neg sat(X, c) \wedge \neg sat(Y, c)$  and for all positions  $i$  such that the variable  $x_i$  appears in  $c$ , we compute  $\sigma = imp(X|i) + imp(Y|i)$  and we set  $Z|k = flip(X|k)$  where  $k$  is*

the position such that  $\sigma$  is maximum. All the variables with no value take the value of  $X$  or  $Y$  with equal probability.

**Definition 4 F&F (Fleurent Ferland) crossover [5]**

Given  $X$  and  $Y$  two parents and  $Z$  the child. For each clause  $c$  such that  $\text{sat}(X, c) \wedge \neg \text{sat}(Y, c)$  (resp.  $\neg \text{sat}(X, c) \wedge \text{sat}(Y, c)$ ) and for all the positions  $i$  such that the variable  $x_i \in c$ ,  $Z|i = X|i$  (resp.  $Z|i = Y|i$ ). All the variables with no value are randomly valued.

Note that CC, CCTM and F&F crossovers differ on the use of the truth values of the clauses induced by the parents. The main ideas are to correct false clauses, to preserve true clauses and to maintain the structure of the assignments. In order to study the characteristics of the different operators, we reduce the algorithm to a sequence of recombination stage on a population with or without using the selection process between two consecutive stages. In order to study the characteristics of the different crossovers, we reduce the algorithm to a sequence of recombination stage on a population with or without using the selection process between two consecutive stages. The tests are realized with a random 3-sat instance (f500.cnf [12]) with 500 variables and a ratio of clauses-to-variables of 4.3.

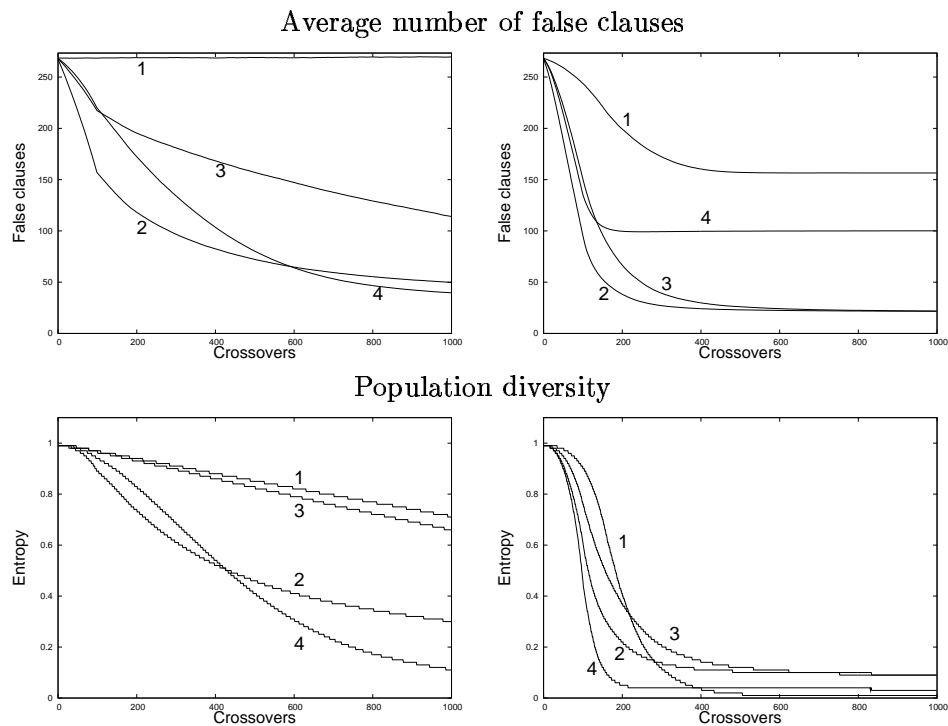


Figure 2: The left column shows the crossovers without the selection stage and the right column shows the crossovers with the selection stage (1 → Uniform crossover, 2 → CCTM crossover, 3 → CC crossover and 4 → F&F crossover)

Figure 2 shows that crossovers behaviors are modified with the addition of the selection process. For the F&F crossover this operator damages its behavior while results are better for the uniform crossover, the CCTM crossover and the CC crossover. The uniform crossover

still does not give good results. Remark that CC and CCTM crossovers give very good results with a diversity higher than all the other crossovers.

An efficient crossover is not necessary a crossover which improves the population enormously but rather should give a good trade-off between quality and diversification of the population. The diversification allows a better exploration of the search space and prevents the population from stagnating in poor local optima.

## 4 Experimental Results with the CC crossover

The GASAT [8] algorithm (figure 1) is a hybrid algorithm using the CC crossover and a local search. GASAT obtains very interesting results on SAT and MAX-SAT problems for large instances (table 1).

Benchmarks				GASAT			Walksat		UnitWalk			
instances	var	cls	sat	%	sec.	fl.	%	sec.	fl.	%	sec.	fl.
hgen2-1205525430	500	1750	Y	5	113.07	2033273						
hgen2-512100147	500	1750	Y		(1 clauses)							
mat25.shuffled	588	1968	N		(3 clauses)							
mat26.shuffled	744	2464	N		(2 clauses)							
color-10-3	300	6475	Y	100	208.33	929366						
color-22-5	2420	272129	?		(10 clauses)							
difp_19_0_arr_rcr	1201	6563	Y		(8 clauses)							
difp_19_99_arr_rcr	1201	6563	Y		(11 clauses)							
g125.18	2250	70163	Y	50	1878.43	149023						
g250.29	7250	454622	Y		(13 clauses)							

Table 1: Comparison of GASAT, Walksat and UnitWalk in the same conditions (if no assignment is found then the best number of false clauses is given)

## References

- [1] Belaid Benhamou and Lakhdar Sais. Theoretical study of symmetries in propositional calculus and applications. In *CADE'92*, pages 281–294, 1992.
- [2] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, Jul 1962.
- [3] Kenneth A. De Jong and William M. Spears. Using genetic algorithm to solve NP-complete problems. In *Proc. of the Third Int. Conf. on Genetic Algorithms*, pages 124–132, San Mateo, CA, 1989.
- [4] Olivier Dubois and Gilles Dequen. A backbone-search heuristic for efficient solving of hard 3-SAT formulae. In Bernhard Nebel, editor, *Proc. of the IJCAI'01*, pages 248–253, San Francisco, CA, 2001.

- [5] Charles Fleurent and Jacques A. Ferland. Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. In *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 619–652, 1994.
- [6] Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman & Company, San Francisco, 1978.
- [7] Jens Gottlieb, Elena Marchiori, and Claudio Rossi. Evolutionary algorithms for the satisfiability problem. *Evolutionary Computation*, 10(1):35–50, 2002.
- [8] Jin-Kao Hao, Frédéric Lardeux, and Frédéric Saubion. Evolutionary computing for the satisfiability problem. In *Applications of Evolutionary Computing*, volume 2611 of *LNCSE*, pages 259–268, University of Essex, England, UK, 14-16 April 2003.
- [9] Edward A. Hirsch and Arist Kojevnikov. UnitWalk: A new SAT solver that uses local search guided by unit clause elimination. PDMI preprint 9/2001, Steklov Institute of Mathematics at St.Petersburg, 2001.
- [10] Chu Min Li. Integrating equivalency reasoning into davis-putnam procedure. In *Proc. of the AAAI'00*, pages 291–296, 2000.
- [11] Chu Min Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proc. of the IJCAI'97*, pages 366–371, 1997.
- [12] David G. Mitchell, Bart Selman, and Hector J. Levesque. Hard and easy distributions for SAT problems. In *Proc. of AAAI'92*, pages 459–465, 1992.
- [13] Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. In *Proc. of the AAAI, Vol. 1*, pages 337–343, 1994.
- [14] Bart Selman, Hector J. Levesque, and David G. Mitchell. A new method for solving hard satisfiability problems. In *Proc. of the AAAI'92*, pages 440–446, San Jose, CA, 1992.
- [15] William M. Spears. Simulated annealing for hard satisfiability problems. In *Second DIMACS implementation challenge : cliques, coloring and satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 533–558, 1996.
- [16] Mutsunori Yagiura and Toshihide Ibarak. Efficient 2 and 3-flip neighborhood search algorithms for the MAX SAT: Experimental evaluation. *Journal of Heuristics*, 7(5):423–442, 2001.
- [17] Hantao Zhang. SATO: An efficient propositional prover. In *Proc. of the 14th International Conference on Automated deduction*, volume 1249 of *LNAI*, pages 272–275, Berlin, 1997.