Solving Strategies using a Hybridization Model for Local Search and Constraint Propagation

Tony Lambert
LERIA, Université de Angers, France
and
LINA, Université de Nantes, France
Tony.Lambert@univ-angers.fr

Eric Monfroy
LINA, Université de Nantes, France
and
Departamento de Informática, Universidad Técnica Federico Santa María, Valparaíso, Chile
Eric.Monfroy@lina.univ-nantes.fr

Frédéric Saubion
LERIA, Université de Angers, France
Frédéric.Saubion@univ-angers.fr

ABSTRACT

Hybridization of local search and constraint programming techniques for solving Constraint Satisfaction Problems is generally restricted to some kind of master-slave combinations for specific classes of problems. In this paper we study combination strategies at a finer hybridization grain, based on a theoretical model for hybridization of local search and constraint propagation. In this framework, hybrid resolution can be achieved as the computation of a fixpoint of some specific reduction functions. Some experimental results show the interest of the model to design such hybridizations.

Categories and Subject Descriptors

I.2.8 [**ARTIFICIAL INTELLIGENCE**]: Problem Solving, Control Methods, and Search—*backtracking, heuristic methods*

key-words: CSP, local search, constraint propagation, hybrid resolution.

1. INTRODUCTION

Constraint Satisfaction Problems (CSP) are usually defined by a set of variables associated to domains of possible values and by a set of constraints. They provide a modeling framework for many computer aided decision making practical applications (such as planning, scheduling, time tabling,...). We only consider here CSP over finite domains.

Solving a CSP consists in finding an assignment of values to the variables that satisfies the constraints. Many resolution algorithms have been proposed and can be classified in two main groups :

- *complete methods* aim at exploring the whole search space in order to find all the solutions or to detect that the CSP is not consistent. Concerning complete resolution techniques methods, we are here mainly concerned by constraint propagation with split (variables domains split) which is one of the most famous techniques of Constraint Programming (CP).

- *incomplete methods* mainly rely on the use of heuristics providing a more efficient exploration of interesting areas of the search space in order to find some solutions. We focus here on local search (LS) techniques.

In order to improve the efficiency of the solving algorithms, combinations of resolution paradigms and techniques have been studied (e.g., [4] presents an overview of possible uses of local search [1] in constraint programming [3]). The benefit of the hybridization LS+CP is well-known and does not have to be proven (see e.g., [8, 11, 10, 12]). But, most of these works are rather algorithmic approaches which define a kind of master-slave combination (e.g., LS to guide the search in CP, or CP to reduce interesting area of the search space explored in LS), or ad-hoc realizations of systems for specific classes of problems.

In this paper, we are concerned by the modeling of different strategies of local search and constraint propagation hybridizations. To this end, we use the uniform generic hybridization framework [9] which is based on K.R. Apt's chaotic iterations [2], a mathematical framework for iterations of a finite set of functions over "abstract" domains with partial ordering. In this framework, basic hybrid resolution processes (such as domain variable reductions, steps of local search, and enumeration) are considered and managed at the same level by a single mechanism.

In this context, our goal is not to provide competitive results, nor to propose specific problem driven combinations to compete with state of the art solvers, but to illustrate the use of this framework to design hybridization strategies. This allows us to study more precisely the benefit of hybridization at a fine grain level on various basic problems, compared to a single use of the basic resolution techniques.

This paper is organized as follows. In Section 2, we give a brief overview of our framework for hybridization. Strategies are then defined and presented in Section 3. Experimental results are described in Section 4 before concluding in Section 5.

SAC'05 , March 13-17, 2005, Santa Fe, New Mexico, USA

2. THE COMPUTATIONAL FRAMEWORK

We first recall basic notions related to Constraint Satisfaction Problems (CSP). A CSP is a tuple (X, D, C) where $X = \{x_1, \cdots, x_n\}$ is a set of variables taking their values in their respective domains $D = \{D_1, \cdots, D_n\}$. A constraint $c \in C$ is a relation $c \subseteq D_1 \times \cdots \times D_n$.
In order to simplify notations, D will also denote the Cartesian product of D_i and C the union of its constraints. A tuple $d \in D$ is a solution of a CSP (X, D, C) if and only if $\forall c \in C, d \in c$.
Constraint propagation, one of the most famous techniques for solving CSP consists in iteratively reducing domains of variables by removing values that do not satisfy the constraints. However, reduction mechanisms use one or some of the constraints of the CSP. Thus, they enforce a local consistency property (such as arc consistency) but not a global consistency of the CSP. These reductions must be interleaved with a splitting mechanism (such as enumeration) in order to obtain a complete solver, (i.e., a solver which returns only solutions and does not loose any solution).
In [2], K.R. Apt defined a framework for constraint propagation. In [9], the model is extended with splitting operators and local search strategies in order to model the different hybrid solving methods as the computation of a fixpoint of a set of functions on an ordered set.
We can now briefly recall the main lines of this theoretical model.

2.1 Generic Iteration Algorithm

K.R. Apt proposed in [2] a general theoretical framework for modeling such reduction operators. In this context, domain reduction corresponds to the computation of a fixpoint of a set of functions over a partially ordered set. These functions, called *reduction functions*, abstract the notion of constraint. The computation of the least common fixpoint of a set of functions F is achieved by the following algorithm:

GI: Generic Iteration Algorithm
$d := \bot$;
$G := F$;
While $G \neq \emptyset$ do
choose $g \in G$;
$G := G - \{g\}$;
$G := G \cup update(G, g, d)$;
$d := g(d)$;
endwhile

where G is the current set of functions still to be applied ($G \subseteq F$), $d \in D$ a partially ordered set (the domains in case of CSP), and for all G, g, d the set of functions $update(G, g, d)$ from F is such that:

- A : $\{f \in F - G \mid f(d) = d \wedge f(g(d)) \neq g(d)\} \subseteq update(G, g, d)$.
- B : $g(d) = d$ implies that $update(G, g, d) = \emptyset$.
- C : $g(g(d)) \neq g(d)$ implies that $g \in update(G, g, d)$

Suppose that all functions in F are inflationary ($x \sqsubseteq f(x)$ for all x) and monotonic ($x \sqsubseteq y$ implies $f(x) \sqsubseteq f(y)$ for all x, y) and that $(D, \sqsubseteq)$ is finite. Then, every execution of the **GI** algorithm terminates and computes in d the least common fixpoint of the functions from F (see [2]).
This abstract framework can now be extended to the hybridization of resolution techniques.

2.2 Introduction of Local Search

Local search techniques [1] aim at exploring the search space, moving from a sample to one of its neighbors in order to optimize a given criterion. These moves are guided by a fitness function which evaluates the benefit of the configuration in order to reach a local optimum.
In the context of the resolution of a given CSP (X, D, C), the search space can be usually defined as the set of possible tuples of $D = D_1 \times \cdots \times D_n$ and the neighborhood is a mapping $\mathcal{N} : D \rightarrow 2^D$. We may define the set of possible local search paths as:

$$\mathcal{LS} = \bigcup_{i>0} \{p = (s_1, \cdots, s_i) \in D^i \mid \forall j,\ 1 \leq j < i-1,$$

$$s_{j+1} \in \mathcal{N}(s_j) \text{ and } s_1 \in D\}$$

with $D^i = D \times D^{i-1}$, search space of i size paths, since the fundamental property of local search relies on its exploration based on the neighborhood relation.
From a practical point of view, a local search is limited to finite paths according to a stop criterion which can be a fixed maximum number of iterations. We consider an evaluation function $eval : D \rightarrow \mathbb{N}$ such that $eval(s)$ represents the number of constraints unsatisfied by s and $eval(s)$ is equal to 0 iff s is a solution. We have obviously to handle a minimization problem. We denote $s <_{eval} s'$ the fact that $eval(s) < eval(s')$. Therefore, from a LS point of view, a result is either a search path p leading to a solution or a search path of a maximum given size and we introduce the following order on $\mathcal{LS}$.

Definition 1. *We consider an order $\sqsubseteq_{ls}$ on $\mathcal{LS}$ defined by: $p \sqsubseteq_{ls} p'$ with $p = (s_1, \ldots, s_n)$ and $p' = (s'_1, \ldots, s'_m)$ iff $eval(s'_m) = 0$ or $m \geq n$.*

We have now to define the computation structure on which reduction functions will be applied and which includes the new component corresponding to the introduction of LS.

2.3 Computation Structure

In order to handle the different data structures associated to each side of the resolution, we add a particular sample of the search space to each CSP, on which LS will be processed. This sample corresponds indeed to a LS path.

Definition 2. *A CSP with sample (sCSP) is defined by a triple (D, C, p) where*

- $D = D_1, ..., D_n$
- $\forall c \in C, c \subseteq D_1 \times \cdots \times D_n$
- $p \in \mathcal{LS}$

Note that, in the definition, the local search path p should be included in the box defined by D. $SCSP$ is the set of all possible sCSP. We may define now an ordered structure $(SCSP, \sqsubseteq)$.

Definition 3. *Given two sCSPs $\psi = (D, C, p)$ and $\psi' = (D', C, p')$,*

$$\psi \sqsubseteq \psi' \quad \textit{iff} \quad D' \subseteq D \textit{ or } (D' = D \textit{ and } p \sqsubseteq_{ls} p').$$

This relation is extended on 2^{SCSP} *as:*

$$\{\phi_1;...;\phi_k\} \sqsubseteq \{\psi_1;...;\psi_l\},$$
$$\textit{iff } \forall\phi_i, (\exists\psi_j,\ \phi_i \sqsubseteq \psi_j \textit{ and } \nexists\psi_j,\ \psi_j \sqsubset \phi_i)$$

where $i \in [1..k], j \in [1..l]$.

We denote ΣCSP the set 2^{SCSP} which constitutes the key set of the computation structure. We use here σCSP to denote an element of ΣCSP. The least element $\perp$ is $\{(D, C, p)\}$, i.e., the initial σCSP to be solved.
We should note that the notion of solution is clear from each side of the resolution (i.e., LS and CP). From the CP point of view, a solution of a CSP is a tuple which satisfies all the constraints. From the LS point of view, the notion of solution is related to the evaluation function which defines a solution as an element $s \in p$ such that $eval(s) = 0$.
Given a sCSP $\psi = (D, C, p)$, these two points of view induce two sets of solutions $Sol_D(\psi) = \{d \in D | \forall c \in C,\ d \in c\}$ and $Sol_{\mathcal{LS}}(\psi) = \{p = (s_1, \cdots, s_k) \in \mathcal{LS} \,|\, eval(s_k) = 0\}$.

Definition 4. *Given a sCSP* $\psi = (D, C, p)$*, the set of solutions of* ψ *is defined by:*

$$Sol(\psi) = \{(d, C, p) | d \in Sol_D(\psi) \textit{ or } p \in Sol_{\mathcal{LS}}(\psi)\}$$

This notion is extended to any σCSP Ψ *as :* $Sol(\Psi) = \bigcup_{\psi\in\Psi} Sol(\psi)$.

At this step, we have to define the reduction functions on ΣCSP in order to describe the different components of the resolution process : constraint propagation by domain reduction and splitting, combined with local search.

2.4 Functions definitions and properties

Given an element $\Psi = \{\psi_1, \cdots, \psi_n\}$ of ΣCSP, we have to apply functions on Ψ which correspond to domain reduction, domain splitting, and local search. These functions may operate on elements ψ_i of Ψ, and for each ψ_i on some of its components. We should note that since we consider here finite initial CSPs, the structure is a finite partial ordering. The following definitions introduce the fundamental properties of the different operators and their general purpose.

Definition 5 (Domain reduction function). *A domain reduction function is a function red on* ΣCSP *s.t. for all* $\Psi = \{\psi_1, \ldots, \psi_n\} \in \Sigma CSP$, $red(\Psi) = \{\psi'_1, \ldots, \psi'_n\}$ *and* $\forall i \in [1 \cdots n]$*:*

- *either* $\psi_i = \psi'_i$
- *or* $\psi_i = (D, C, p)$, $\psi'_i = (D', C, p)$ *and* $D \supseteq D'$ *and* $Sol(\psi_i) = Sol(\psi'_i)$.

Note that this condition ensures that $\Psi \sqsubseteq red(\Psi)$ and that the function is inflationary and monotonic on $(\Sigma CSP, \sqsubseteq)$. From a constraint programming point of view, no solution of the initial σCSP is lost by a domain reduction function. This is also the case for domain splitting as defined below.

Definition 6 (Domain splitting). *A domain splitting function is a function sp on* ΣCSP *such that for all* $\Psi = \{\psi_1, \ldots, \psi_n\} \in \Sigma CSP$*:*

a. $sp(\Psi) = \{\psi'_1, \ldots, \psi'_m\}$ *with* $n \leq m$,

b. $\forall i \in [1..n]$,

- *either* $\exists j \in [1..m]$ *such that* $\psi_i = \psi'_j$
- *or there exist* $\psi'_{j_1}, \ldots, \psi'_{j_h}$, $j_1, \ldots, j_h \in [1..m]$ *such that* $Sol(\psi_i) = \bigcup_{k=1..h} Sol(\psi'_{j_k})$.

c. and, $\forall j \in [1..m]$,

- *either* $\exists i \in [1..n]$ *such that* $\psi_i = \psi'_j$
- *or* $\psi'_j = (D', C, p)$ *and there exists* $\psi_i = (D, C, p)$, $i \in [1..n]$ *such that* $D \supseteq D'$.

Conditions a. and b. ensure that some sCSPs have been split into sub-sCSPs by splitting their domains (one or several variable domains) into smaller domains without discarding solutions (defined by the union of solutions of the ψ_i). Condition c. ensures that the search space does not grow: none of the domains of the SCSPs composing Ψ' is not included in one of the domain of some SCSP composing Ψ.

Definition 7 (Local Search). *A local search function* λ_N *is a function :*

$$\lambda_N : \Sigma CSP \to \Sigma CSP\ ,$$
$$\{\psi_1, \cdots, \psi_n\} \mapsto \{\psi'_1, \cdots, \psi'_n\}$$

where

- N *is the maximum number of consecutive moves*
- $\forall i \in [1..n]$
  - *either* $\psi_i = \psi'_i$
  - *or* $\psi_i = (D, C, p)$ *and* $\psi'_i = (D, C, p')$ *with* $p = (s_1, \cdots, s_k)$ *and* $p' = (s_1, \cdots, s_k, s_{k+1})$ *such that* $s_{k+1} \in \mathcal{N}(s_k) \cap D$ *and* $k + 1 \leq N$.

N represents the maximum length of a local search path, i.e., the number of moves allowed in a usual local search process. Note that $\psi_i = \psi'_i$ can happen when:

1. $p \in Sol_{\mathcal{LS}}(\psi)$: the last sample s_n of the current local search path cannot be improved using λ_N,
2. $n = N$: the maximum allowed number of moves has been reached,
3. λ_N is the identity function on ψ_i, i.e., λ_N does not try to improve the local search path of the SCSP ψ_i. This might happen when no possible move can be performed (e.g., a descent algorithm has reached a local optimum or all neighbors are tabu in a tabu algorithm [6]).

3. STRATEGIES

From an operational point of view, domain reduction and splitting functions have to be applied on a selected domain of a selected sCSP of a given σCSP. Therefore, a reduction function will be a function :

$$f^{sel_{SCSP}, sel_D} : \Sigma CSP \to \Sigma CSP$$

where :

$$sel_{SCSP} : \Sigma CSP \to SCSP$$

and :

$$sel_D : SCSP \to D$$

Given a σCSP Ψ,

$$f^{sel_{SCSP},sel_D}(\Psi) = f(sel_D(sel_{SCSP}(\Psi)))$$

Concerning local search, the function has to be applied on a chosen sCSP using a specific strategy which provides the next configuration to add to the current path (i.e., the move to be performed). Therefore, a LS function will be a function $l^{sel_{SCSP},strat}$ where :

$$strat\colon SCSP \rightarrow D$$

and such that

$$l^{sel_{SCSP},strat}(\Psi) = (D, C, (s_0, \cdots, s_k, s_{k+1}))$$

with

$$sel_{SCSP}(\Psi) = (D, C, (s_0, \cdots, s_k)$$

and

$$strat(sel_{SCSP}(\Psi)) = s_{k+1}$$

Note that, to simplify the presentation, selection functions have been limited to a single sCSP, a single domain, and a single chosen neighbor but can be extended to more general selection functions [9].
Therefore, these selection functions will be associated to reduction functions in order to model various strategies.
We first introduce classic resolution strategies by defining specific domain and sCSP selection functions as follows:

- Random : Choose any function in the set, consider :

$$rand_D : SCSP \rightarrow D,$$

$$(D_1, \ldots, D_n, C, p) \mapsto D_l$$

with $l \in [1..n]$ and :

$$rand_{SCSP} : \Sigma CSP \rightarrow SCSP,$$

$$\{\psi^1, \ldots, \psi^m\} \mapsto \psi^k$$

with $k \in [1..m]$

- Depth-First: consists in selecting the smallest domain of the smallest $SCSP$, consider :

$$min_D : SCSP \rightarrow D,$$

$$(D_1, \ldots, D_n, C, p) \mapsto D_l,$$

with $l \in [1..n], \forall i \in [1..n], |D_i| \geq |D_l|$ and :

$$min_{SCSP} : \Sigma CSP \rightarrow SCSP,$$

$$\{\psi^1, \ldots, \psi^m\} \mapsto \psi^k,$$

with $k \in [1..m], \forall i \in [1..m], |min_D(\psi^i)| \geq |min_D(\psi^k)|$

- Width-First: consists in selecting the biggest domain of the biggest $SCSP$, consider :

$$max_D : SCSP \rightarrow D,$$

$$(D_1, \ldots, D_n, C, p) \mapsto D_l,$$

with $l \in [1..n], \forall i \in [1..n], |D_i| \leq |D_l|$ and :

$$max_{SCSP} : \Sigma CSP \rightarrow SCSP,$$

$$\{\psi^1, \ldots, \psi^m\} \mapsto \psi^k$$

with $k \in [1..m], \forall i \in [1..m], |max_D(\psi^i)| \leq |max_D(\psi^k)|$

- LS-Forward Checking : standard forward checking consists in instantiating variables in a given order (e.g., variable index) and to prevent future conflicts by applying reduction functions on the current variable and on those which are not yet instantiated. This mechanism restricts reductions because forward checking checks only the constraints between the current variable and the next variables. We extend forward checking to take LS functions into account. As a consequence, we have specific selection functions to parameterize our reduction (red), split (sp) and local search (λ).

$$f^{FC_{SCSP},FC_D} : \Sigma CSP \rightarrow \Sigma CSP,$$

  - If $f \in \{red; \lambda\}$ then :

$$FC_D : SCSP \mapsto D,$$

$$(D_1, \ldots, D_n, C, p) \mapsto D_l,$$

$\forall i \in [1..l], |D_i| = 1$ and $|D_{l+1}| \neq 1$

$$FC_{SCSP} : \Sigma CSP \rightarrow SCSP,$$

$$\{\psi^1, \ldots, \psi^m\} \mapsto \psi^k$$

with $k \in [1..m]$ and $\forall j \in [1..m] |FC_D(\psi^j)| \leq |FC_D(\psi^k)|$

  - Else $f = sp$ and :

$$FC_D : SCSP \rightarrow D,$$

$$(D_1, \ldots, D_n, C, p) \mapsto D_l$$

such as $\forall i \in [1..l-1], |D_i| = 1$ and $|D_l| \neq 1$

$$sp^{FC_{SCSP},FC_D} : \Sigma CSP \rightarrow \Sigma CSP,$$

$$\{\psi^1, \ldots, \psi^n\} \mapsto \{\psi^1, \ldots, \psi^{k,1}, \ldots, \psi^{k,t}, \ldots, \psi^n\}$$

such as $FC_{SCSP}(\{\psi^1, \ldots, \psi^n\}) = \psi^k$, $|FC_D(\psi^k)| = t$

4. EXPERIMENTATION

In this section, we present a prototype implementation, developed in C++, which allows us to test hybridizations on different CSP example. The purpose of this section is not to test a high performance algorithm on large scale benchmarks but to highlight the benefit of hybrid resolution over various problems.

4.1 Experimental Process

Our basic functions are organized into three sets: a first set of domain reduction functions dr, the set of split functions sp, and the set of LS functions ls. According to the generic algorithm **GI**, one has to define a choose strategy and to update the sets of functions at each iteration. This is described here by a triple (α, β, γ) such that α, β, γ represents the percentage of reduction functions, split functions, and local search functions respectively, that are applied. Then, functions are fairly selected in the sets with respect to these ratios.

Instantiating Sets of Functions
We propose here different sets of functions corresponding to different strategies as described in the previous section. A strategy corresponds actually to the instantiation of the three sets (dr, sp, and ls) mentioned above by the our basic types of functions (red, sp, and λ). Here, red corresponds to bound consistency operators and global constraints (e.g., $alldifferent$). sp is a split operator which cuts the selected domain into two subdomains and λ is a basic LS move which will be instantiated by a tabu search strategy which selects the best configuration which is not in a tabu list as the next element of the path. Tabu list length is set to 10.

Random
$dr_{Rand} = \{f^{rand_{SCSP}, rand_D} : \Sigma CSP \rightarrow \Sigma CSP \mid f \in red\}$
$sp_{Rand} = \{f^{rand_{SCSP}, rand_D} : \Sigma CSP \rightarrow \Sigma CSP \mid f \in sp\}$
$ls_{Rand} = \{f^{rand_{SCSP}, tabu} : \Sigma CSP \rightarrow \Sigma CSP \mid f \in \lambda\}$

Depth First
$dr_{DF} = \{f^{min_{SCSP}, min_D} : \Sigma CSP \rightarrow \Sigma CSP \mid f \in red\}$
$sp_{DF} = \{f^{min_{SCSP}, min_D} : \Sigma CSP \rightarrow \Sigma CSP \mid f \in sp\}$
$ls_{DF} = \{f^{min_{SCSP}, tabu} : \Sigma CSP \rightarrow \Sigma CSP \mid f \in \lambda\}$

Width First
$dr_{WF} = \{f^{max_{SCSP}, max_D} : \Sigma CSP \rightarrow \Sigma CSP \mid f \in red\}$
$sp_{WF} = \{f^{max_{SCSP}, max_D} : \Sigma CSP \rightarrow \Sigma CSP \mid f \in sp\}$
$ls_{WF} = \{f^{max_{SCSP}, tabu} : \Sigma CSP \rightarrow \Sigma CSP \mid f \in \lambda\}$

LS-Forward Checking
$dr_{FC} = \{f^{FC_{SCSP}, FC_D} : \Sigma CSP \rightarrow \Sigma CSP \mid f \in red\}$
$sp_{FC} = \{f^{FC_{SCSP}, FC_D} : \Sigma CSP \rightarrow \Sigma CSP \mid f \in sp\}$
$ls_{FC} = \{f^{FC_{SCSP}, tabu} : \Sigma CSP \rightarrow \Sigma CSP \mid f \in \lambda\}$

Selected Problems: we consider various classic CSP : *S+M=M* (Send + More = Money), *Magic Square*, *Langford number*, *Zebra*, *Golomb ruler*, and the *Uzbekian problem*, issued from the CSPlib [5].

4.2 Experimental results

The evaluation criterion corresponds to the number of applications of basic functions to reach a first solution. Such an application of function corresponds either to a step of local search, to a split, or to a propagation (reduction of one domain using one constraint). We focus here on small problems and computation times for the following tests represent less than 1 minute of CPU time (for example a solution for the langford number is found in 1 sec.).

Interaction between CP and LS
We study here the benefit of the hybridization of LS and CP using different strategies in order to highlight the effect of cooperation on solving efficiency.

We first focus on the problems S+M=M and Langford Number; tests are performed by growing up the percentage α of propagation. To insure to get a solution, we set the split ratio to $\beta = \alpha * 0.1$. For example, if we set 40% of propagation, we will set 4% of split and thus 56% of LS. These tests use a depth-first strategy.

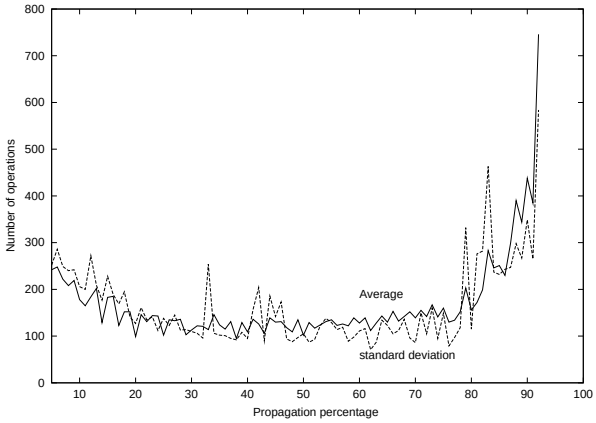

Fig1: Cost of a solution LNumber (Depth-First)

Figure 1 shows that the best result for the Langford number problem corresponds to a range of propagation rate between 35% and 45%. As a matter of fact, when local search represents less then 10% of the search effort, reaching a solution means computing the fixpoint for constraint propagation (and so applying all propagation functions). Note that, for this problem, tabu search alone provides better results than propagation with split.

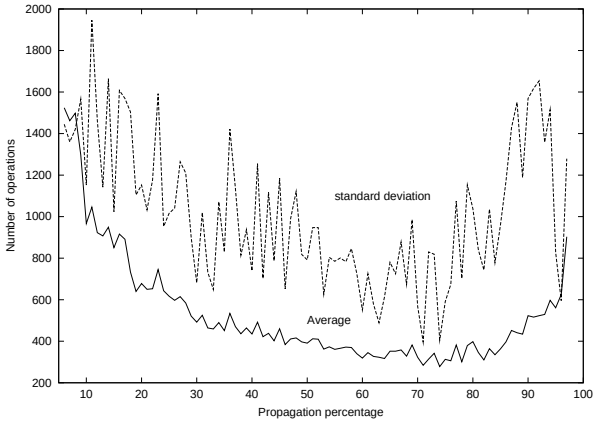

Fig2 : Cost of a solution S+M (Depth-First)

On Figure 2, the depth-first strategy is applied on the S + M = M problem. There is an important standard deviation: indeed, although the sCSP and the domain are selected, the choice of functions to apply is not fixed. But the average curve seems to be more regular, and the best range is 70%–80%. Here, LS alone is less efficient than CP. Thus, choosing the best tuning for hybridization depends on the problem and on the strategies that are applied. Table 1 presents the best ranges using Forward-checking on different problems.
These results point out that the incremental introduction of CP in LS (the same remark is valid for LS in CP) improves the general efficiency of resolution. This hybridization can thus be tuned to optimize performances.

Problem	S+M	LN42	Zebra	M. square	Golomb
Rate FC	70-80	15 - 25	60-70	30-45	30 - 40

Table 1: Best range of propagation rate (α) to compute a solution

Benefit of Hybridization w.r.t. LS and CP alone
We present here a comparative study of hybridization CP+LS which corresponds to:

- the use of the three strategies with ratios (α, β, γ) (best ratios are selected w.r.t. Table 2).
- CP alone, i.e., $(90\%, 10\%, 0)$,
- and LS alone, i.e., $(0, 0, 100\%)$.

These methods are used with the different strategies described above.

Strategy	Method	S+M	LN42	M. square	Golomb
Random	LS	1638	383	3328	3442
	CP+LS	1408	113	892	909
	CP	3006	1680	1031	2170
D-First	LS	1535	401	3145	3265
	CP+LS	396	95	814	815
	CP	1515	746	936	1920
FC	LS	1635	393	3240	3585
	CP+LS	22	192	570	622
	CP	530	425	736	1126

Table 2: Avg. nb. of operations to compute a first solution

These results show that hybridization benefits again from the interaction between the resolution components: this allows us to improve the efficiency w.r.t. resolution steps.
Improvements occur on problems for which LS is better than CP but also on problems for which CP is better than LS. Moreover, the improvement is strongly related to the problem structure (such as the density of solutions) and to the chosen strategy. Experiments with a Width-First strategy are not presented here but provide similar results.

5. PERSPECTIVES AND CONCLUSION

Most of hybrid approaches are ad-hoc algorithms based on a master-slave combination: they favor the development of systems whose efficiency is strongly related to a given class of CSPs. In this paper, we have used a more suitable general framework to model different strategies of combinations in order to highlight the benefit of the hybridization at a fine grain level.
These preliminary results allow us to identify the interaction between the resolution mechanisms and such studies could be used to tune general purpose hybrid solvers.

In the future, we plan to extend the framework in order to handle optimization problems. From a LS point of view, this will change the *strat* functions used to create the reduction functions. From a CP point of view, algorithms such as Branch and Bound requires adding new constraints during resolution: this could be done using a new type of reduction function in the model.
An other future extension is to provide some "tools" to help designing finer strategies in the GI algorithm. To this end, we plan to extend works of [7] where strategies are built using some composition operators in the GI algorithm. Moreover, this will also open possibilities of concurrent and parallel application of reduction functions inside the model.

6. ACKNOWLEDGMENTS

We would like to thank the anonymous referees for their helpful comments and remarks. This work is supported by the CPER COM project.

7. REFERENCES

[1] E. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., 1997.

[2] K. R. Apt. From chaotic iteration to constraint propagation. In *Proceedings of the 24th International Colloquium on Automata, Languages and Programming*, pages 36–55. Springer-Verlag, 1997.

[3] K. R. Apt. *Principles of Constraint Programming*. Cambridge University Press, 2003.

[4] F. Focacci, F. Laburthe, and A. Lodi. Local search and constraint programming. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research and Management Science*. Kluwer Academic Publisherss, Norwell, MA, 2002.

[5] I. Gent, T. Walsh, and B. Selman. http://www.4c.ucc.ie/ tw/csplib/, funded by the UK Network of Constraints.

[6] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[7] L. Granvilliers and E. Monfroy. Implementing constraint propagation by composition of reductions. In *Logic Programming, 19th International Conference, ICLP*, volume 2916 of *Lecture Notes in Computer Science*, pages 300–314. Springer, 2003.

[8] N. Jussien and O. Lhomme. Local search with constraint propagation and conflict-based heuristics. *Artificial Intelligence*, 139(1):21–45, 2002.

[9] E. Monfroy, F. Saubion, and T. Lambert. On hybridization of local search and constraint propagation. In *Proceedings of ICLP'2004*, volume 3132 of *LNCS*, pages 299–313. Springer Verlag, 2004.

[10] G. Pesant and M. Gendreau. A view of local search in constraint programming. In E. C. Freuder, editor, *Principles and Practice of Constraint Programming*, volume 1118 of *Lecture Notes in Computer Science*, pages 353–366. Springer, 1996.

[11] S. Prestwich. A hybrid search architecture applied to hard random 3-SAT and low-autocorrelation binary sequences. In *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming*, pages 337–352. Springer-Verlag, 2000.

[12] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*, volume 1520, pages 417–431. Springer-Verlag, 1998.