Deriving information from sampling and diving

Michele Lombardi [C]
michele.lombardi2@unibo.it
DEIS, Alma Mater Studiorum *Università di Bologna*
V.le Risorgimento 2, 40136, Bologna, Italy

Andrea Roli
andrea.roli@unibo.it
DEIS - Campus of Cesena, Alma Mater Studiorum
Università di Bologna
Via Venezia 52, 47023, Cesena, Italy

Michela Milano
michela.milano@unibo.it
DEIS, Alma Mater Studiorum *Università di Bologna*
V.le Risorgimento 2, 40136, Bologna, Italy

Alessandro Zanarini
alessandro.zanarini@dynadec.com
Dynadec Europe
Place de l'Université 16, 1348, Louvain-La-Neuve, Belgium

Keywords: Constraint Satisfaction Problem, Sampling, Diving, Solution Counting, Heuristics Evaluation

Abstract. We investigate the impact of information extracted from *sampling* and *diving* on the solution of Constraint Satisfaction Problems (CSP). A sample is a complete assignment of variables to values taken from their domain according to a given distribution. Diving consists in repeatedly performing depth first search attempts with random variable and value selection, constraint propagation enabled and backtracking disabled; each attempt is called a dive and, unless a feasible solution is found, it is a partial assignment of variables (whereas a sample is a –possibly infeasible– complete assignment). While the probability of finding a feasible solution via sampling or diving is negligible if the problem is difficult enough, samples and dives are very fast to generate and, intuitively, even when they are infeasible, they can provide some statistical information on search space structure. The aim of this paper is to understand to what extent it is possible to support the CSP solving process with information derived from sampling and diving. In particular, we are interested in extracting from samples and dives precise indications on the quality of individual variable-value assignments with respect to feasibility. We formally prove that even uniform sampling could provide precise evaluation of the quality of single variable-value assignments; as expected, this requires huge sample sizes and is therefore not useful in practice. On the contrary, diving is much better suited for assignment evaluation purposes. We undertake a thorough experimental analysis on a collection of Partial Latin Square and Car Sequencing instances to assess the quality of information provided by dives. Dive features are identified and their impact on search is evaluated. Results show that diving provides information that can be fruitfully exploited.

[C]Corresponding author

1. Introduction

Exploiting information collected during search is often one of the key components for the successful solution of Constraint Satisfaction Problems (CSPs). In this work, we address the issue of formally studying to what extent it is possible to extract information from two kinds of sampling techniques, namely random sampling and diving. In particular, we are interested in extracting precise indications on how good/bad are individual variable-value assignments, with a focus on information regarding problem feasibility.

The collected information can be used for designing search heuristics, for partitioning domains in decomposition based search [10], for over-filtering unpromising values from variable domains, or for defining suitable local moves to repair an infeasible assignment.

In this paper, we will call a *sample* an assignment of variables to values taken from their domains. In case of random samples, variables and values are chosen according to a uniform distribution. A *dive*, on the contrary, is obtained by tree search in which a variable and a value in its domain are randomly chosen and after each assignment propagation filters out values proved to be infeasible with the current partial assignments. We will limit our discussion to samples and dives collected at the root node (in a pre-processing phase of the search) and to uniform distributions.

We will first define a formal probabilistic model that makes it possible to estimate through sampling the probability that the problem remains feasible after a specific event occurs, such as a single variable-value assignment. This study shows that it is possible to extract valuable information from random sampling in a sound theoretical framework. Nevertheless, in the case of large-size instances, this approach has the drawback that a good estimation of the variable-value assignment probability can be achieved only at the price of huge sample sizes.

As a second step, we will show how to extract dive features to measure the potential feasibility and infeasibility of single variable-value assignments, using Partial Latin Square completion and Car Sequencing as sample problems [8]. Parameters such as the average dive length and the number of variable-value occurrences enable us to extract pieces of information on the quality of single variable-value assignments. Experimental results show that some of the features extracted from dives are very informative, having accuracy (w.r.t. to feasibility) comparable to or better than a state-of-the-art heuristic used in Impact Based Search [17].

To the best of our knowledge, a formal analysis of this kind has not yet been published in the literature, except for our introductory work on this subject [14]. This paper substantially extends and completes our previous work by adding further experimental evaluations mainly concerning *(i)* the correlation between the real solution density (computed by exhaustive enumeration) and parameters derived from diving by defining a set of error rate indicators; *(ii)* the complementarity of the diving parameters; *(iii)* the effect of different propagation algorithms on diving parameters; *(iv)* the use of diving parameters into a search strategy and their comparison with impact based search; *(v)* experimentations on a new problem domain, that is the Car Sequencing problem.

The remainder of the paper is organized as follows: in Sect.2, we provide an overview of related work. In Sect.3, we present a formal model for evaluating the quality of information provided by sampling and delineating the limits of its applicability. In Sect.4, we define diving and the main features which will be considered in the experimental analysis presented in Sect.5. Finally, in Sect.6 we summarize the main results of this contribution and we outline directions for further research.

2. Related work

Uniform sampling of solutions has been recently studied by Gogate and Dechter in [4]. Their approach uses a randomized backtracking procedure guided by a heuristic computed through generalized belief propagation. In order to avoid the rejection problem (ending up to an infeasible solution) they systematically search for a feasible solution using a standard backtracking procedure. In a follow-up work [5] they added a second re-sampling phase that gives approximation guarantees that the distribution of sampled solutions converges to the uniform distribution.

Gomes et al. in [7] also propose an algorithm for sampling solutions of SAT problems. They iteratively add streamlining constraints to the SAT formula to reduce the set of solutions (possibly to only one) and they return one solution chosen uniformly at random. Our approach mainly differs from the two described above in the fact that we sample assignments that can be either (very unlikely) feasible or infeasible whereas they sample only feasible solutions (thus with an exponential computational effort).

Knuth proposed in 1975 [12] an estimator of a search tree based on random probing of the search space, exploiting the branching factor at each node of the search tree.

In [2], Boussemart et al. propose a conflict-driven variable ordering heuristic: they extend the concept of variable degree integrating a simple but effective learning technique that takes into account failures. Basically, each constraint has a weight associated that is increased by one each time the constraint leads to a failure (i.e. a domain wipe-out). Grimes and Wallace in [9] employ random dives (referred to as probes) to initialize the process. Nevertheless, during the diving phase, they do not collect any explicit information regarding variable assignments and dive lengths.

Ruml proposes in [20] to use adaptive probes to iteratively learn the search strategy to be employed in exploring a search tree. In constraint satisfaction problems, he measures the quality of a dive in terms of the number of instantiated variables (dive length). The algorithm then adaptively adjusts the heuristic used for the following dives.

In [1] random dives are used to initialize an elite set of partial solutions where the number of assigned variables is taken as a quality measure. An elite solution is then randomly chosen to provide heuristic guidance. Note that the heuristic is based solely on a single elite (partial) solution whereas our approach tries to infer general information from the whole set of partial solutions.

Refalo in [17] proposes a heuristic (known as Impact Based Search) that estimates the effort of the search as the proportional reduction of the Cartesian product of the variable domains after a variable instantiation. This estimation (referred to as *impact*) is initialized at the beginning of the search through a dichotomic search in which each variable domain is iteratively split into two sub-domains.

As an interesting connection, in the following we cite some approaches that share the same aim of this paper, i.e., deriving measures on the quality of individual variable-value assignments with respect to feasibility. Hsu et al. [11] apply a Belief Propagation algorithm within an Expectation Maximization framework (EMBP) in order to approximate variable biases (or marginals), i.e., the probability that a variable takes a given value in a solution. Even though the approach is general, they derive formulas for computing and updating the variable biases only for the `alldifferent` constraint and they experimented on the Partial Latin Square problem. The computation of variable biases is time-consuming but they show that the heuristic is effective even when it is employed only in the top part of the search tree. That work has been recently extended by Le Bras et al. in [13]; the contribution is twofold: firstly, they generalize Hsu et al.'s method to tackle any global constraint; secondly, they leverage the same method to consider the problem model as an individual (likely NP-Hard) constraint. Unfortunately, previous works do not

assess the quality of the derived estimates w.r.t. exact probabilities.

Finally, Zanarini et al. [21] employs counting algorithm at the constraint level to find an exact or approximate probability of a given variable-value pair to be in a solution of a constraint. The estimates are exploited to guide the search and they attain promising results on some benchmark problems. Nevertheless, the information extracted by this approach is focused on individual constraints rather than on the whole problem as we do in this study.

We point out that in none of the methods described above systematic statistic measures are extracted from the random dives; in fact, our approach can be thought as being orthogonal to the described ones, as the emphasis is on evaluating the feasibility related information extracted from samples and dives rather than on providing a top quality heuristic. As a consequence, the method we propose can be easily integrated in most of the mentioned frameworks.

3. Uniform Sampling

In this section, we study the quality of information that samples can provide on individual variable-value assignments. A formal model is defined in terms of probability that the problem at hand remains feasible after a variable-value assignment. This analytical model makes it possible to formally express the information provided by sampling and the relation between number of samples and information accuracy.

Definition 3.1. Assume we are given a $CSP = (X, D, C)$ where $X = \{X_1, \dots X_n\}$ is the set of variables, $D = \{D_1, \dots D_n\}$ the set of domains and $C = \{c_1, \dots c_k\}$ the set of constraints. A *sample* s is an element of the Cartesian product of the domains of all problem variables, $s \in \prod_{i=1}^{n} D_i$. Constraints are not taken into account in the definition of s. A *uniform*, or *random*, *sample* is a sample taken w.r.t. a uniform distribution over $\prod_{i=1}^{n} D_i$.

The idea is that by sampling the search space randomly, either we find a solution to the problem, and we are done, or we collect only infeasible samples. In the second case, we can try to measure the feasibility or infeasibility degree of single variable-value assignments. Suppose to model the sample generation problem as a stochastic variable with sample space $\Omega = \prod_i D_i$. In this context the set F of feasible solutions and the set of infeasible solutions I are events, such that $F \cup I = \Omega$, $F \cap I = \emptyset$ and $P(F)$ (resp. $P(I)$) is the probability of a random sample to be feasible (resp. infeasible).

Now, let $A_0, A_1, \dots A_m$ be a set of events we want to evaluate; one possible event can be a single variable-value assignment, another can be the removal of a value from a domain. We want to estimate the probability the problem is still feasible after the event occurs. For example, let us consider the problem of choosing a value v from the domain of a variable X_i, as it would be done by a value selection heuristic; we want the problem to remain feasible after the assignment. More formally, given a threshold value $\theta \in [0, 1)$, we want $P(F \mid X_i = v) > \theta$; that is, we want the probability to randomly pick a feasible solution containing $X_i = v$ to be greater than the specified threshold. As in this work we are mostly concerned with problem feasibility, a single variable assignment $X_i = v$ is considered better than $X_j = u$ if it has a greater value for the probability $P(F \mid X_i = v)$. So, we need a way to estimate the conditional probability $P(F \mid X_i = v)$: this can be done by using probability theory results [15]. From

the Bayes theorem we know that:

$$P(F \mid X_i = v) = \frac{P(F)P(X_i = v \mid F)}{P(X_i = v)}$$

where $P(X_i = v \mid F)$ is the probability that $X_i = v$ in a random *feasible* solution. Probability $P(X_i = v)$, instead, is defined on the entire sample space and therefore $P(X_i = v) = 1/|D_i|$, where $|D_i|$ is the cardinality of the domain of X_i. Therefore, in order to identify good assignments we only need a way to compute $P(F)P(X_i = v \mid F)$. For doing this, we consider another result in probability theory which states that, given a set of events $B_0, B_1, \ldots$ such that $B_i \cap B_j = \emptyset$ for each $i \neq j$ and $\bigcup_i B_i = \Omega$, and an arbitrary event A:

$$P(A) = \sum_j P(B_j)P(A \mid B_j)$$

The event A in our case is the assignment $X_i = v$, while events B_j are F and I, that have empty intersection:

$$P(X_i = v) = P(F)P(X_i = v \mid F) + P(I)P(X_i = v \mid I)$$

hence:

$$P(F)P(X_i = v \mid F) = P(X_i = v) - P(I)P(X_i = v \mid I)$$

Note that the probability $P(X_i = v \mid I)$ can be estimated by generating a number of infeasible solutions and counting the occurrences of the $X_i = v$ assignment. Indeed, since in principle we randomly sample the whole space $F \cup I$, also I is randomly sampled. Thus, by counting the frequency of variable-value assignments restricted to the infeasible samples, by definition of conditional probability, we do estimate $P(X_i = v \mid I)$. Therefore:

$$\begin{aligned} P(F \mid X_i = v) &= \frac{P(F)P(X_i = v \mid F)}{P(X_i = v)} \\ &= \frac{P(X_i = v) - P(I)P(X_i = v \mid I)}{P(X_i = v)} \\ &= 1 - P(I)\frac{1}{P(X_i = v)}P(X_i = v \mid I) \end{aligned}$$

Finally, since $P(X_i = v) = 1/|D_i|$:

$$P(F \mid X_i = v) = 1 - P(I)\ \ |D_i|\ \ P(X_i = v \mid I) \tag{1}$$

where $P(I)$ is an unknown, problem dependent, constant factor which has no contribution when used to compare assignments. Note that domain size does influence the evaluation of the assignments. Formula (1) provides us with a way to evaluate assignments of single values to single variables; the result can be generalized to take into account a generic event A and thus target other search strategies as well.

Unfortunately, formula (1) is useless in most practical cases: this is due to the sample size needed to get good probability approximations, especially when there are few feasible solutions. The sample size can be computed with the following formula, used to bound the absolute error on the fraction of a population (for example our solutions) satisfying a specific condition (for example $X_i = v$):

$$n = z_\alpha \frac{p(1-p)}{\delta^2}$$

where p is a rough estimate of the probability that a random chosen solution satisfies the condition, δ is the absolute error we allow and the z_α parameter depends on the confidence level we want to achieve. For example in a Partial Latin Square instance of order 25, with around 2000 feasible solutions and average domain size of 4, in order to get a confidence interval of 10% with a 95% confidence level we need the following sample size:

$$n = 4 \times \frac{P(X_i = v|I)P(X_i \neq v|I)}{(0.1 \times 2000/25^{25})^2} \simeq 4 \times \frac{0.25 \times 0.75}{(0.1 \times 2000/25^{25})^2} \simeq 1.48 \times 10^{65}$$

where 4 is the value of the z_α parameter for a 95% confidence level, $P(X_i = v|I)$ is estimated to be roughly 0.25 since the average domain size is 4, we want an absolute error of around 10% (0.1) and the order of magnitude of the probability of a feasible solution is $(2000/25^{25})$.

Although not often applicable in most of the practical cases, formula (1) is based on a valid probabilistic model which can be used to draw insightful conclusions. First of all, we proved that a relation exists between the occurrence of a single assignment $X_i = v$ in *infeasible* solutions and in *feasible* solutions; this fact could sound counterintuitive, because at a first sight one might think that infeasible assignments do not provide any piece of information. Second, we proved that domain size influences the choice of the best variable.

In the next section, we introduce diving as an alternative way of extracting information. We will see that statistics computed over features of dives, which can be collected far more efficiently than samples, provide high quality information to the search process.

4. Diving

The main difference between dives and samples is that, in the construction of dives, constraints are taken into account. In the literature, dives are often referred to as probes. Intuitively, a dive collects variable assignments (derived both from branching decisions and from propagation) up to a failure. We do not include the choice that causes failure, so that the dive is in general a consistent (depending on the chosen propagation algorithms) partial assignment.

First, we have to clarify when we consider that a constraint c_j is consistent after some variables are assigned. Suppose we have a set of variables $X_1, \ldots, X_n$ with domains $D_1, \ldots D_n$. A partial assignment on a subset of m variables $\gamma = (X_{i1} = v_{i1}, \ldots, X_{im} = v_{im})$ is *consistent* if all the constraints mentioning assigned variables $X_{i1}, \ldots X_{im}$ are consistent. A constraint c_j is consistent if, after propagating the effects of the assignment, the not yet instantiated variables appearing in the constraints have a non empty domain. We refer to the constraint c_j subject to the assignment γ to as $c_j|_\gamma$. Clearly, some assigned variables may not appear in the constraint. In this case the assignment has no effect. More formally:

Definition 4.1. Given a $CSP = (X, D, C)$ where $X = \{X_1, \ldots X_n\}$ is the set of variables, $D = \{D_1, \ldots D_n\}$ the set of domains and $C = \{c_1, \ldots c_k\}$ the set of constraints. A *dive* γ is an assignment of (some) variables to values $(X_{i1} = v_{i1}, \ldots X_{im} = v_{im})$ where $m \leq n$ such that for each problem constraint c_j, $c_j|_\gamma$ is consistent.[1]

[1] Any form of local consistency can be considered, although it is likely to have an impact both on the quality of the information extracted and on the performance of the dive generation.

For example, let be given a problem with ten variables $X_1, \dots X_{10}$. The first four variables X_1, X_2, X_3, X_4 have a domain containing values from 1 to 4, while the remaining variables $X_5, \dots X_{10}$ have a domain from 1 to 10. Suppose we have a constraint $c_j = \texttt{alldifferent}([X_1, X_2, X_3, X_4])$, for which GAC in enforced; c_j is consistent with the assignment $X_1 = 1, X_2 = 2, X_5 = 7, X_7 = 8$ since, after propagation, values 3 and 4 are left in X_3 and X_4 domains. We refer to this instantiated constraint as $c_j|_{\{X_1=1, X_2=2, X_5=7, X_7=8\}}$. Clearly, the assignments $X_5 = 7, X_7 = 8$ have no effect on the constraint.

In our experiments, we consider a special case of dives built as follows. We visit the search tree by choosing randomly at each node a variable and choosing randomly a value in its domain. Propagation is activated, so as to remove inconsistent values from variable domains. Possibly, some variables become instantiated due to propagation. As soon as a failure occurs, we retract the last assignment and we maintain the consistent set of assignments up to that point. Hence, assignments in a dive are derived both by branching choices and by propagation.

For example, let us consider again the problem with ten variables $X_1, \dots X_{10}$. Now the first three variables X_1, X_2, X_3, have a domain containing values from 1 to 4, variable X_4 has a domain containing values 3 and 4, while the remaining variables $X_5, \dots X_{10}$ have a domain from 1 to 10. Suppose again we have an `alldifferent` constraint $c_1 = \texttt{alldifferent}([X_1, X_2, X_3, X_4])$ (with GAC) and other problem constraints $c_2, \dots c_k$. Suppose we perform tree search by randomly selecting variables and values in this order $X_1 = 4, X_2 = 2, X_5 = 7, X_7 = 8$. After the assignment $X_1 = 4$, the propagation of the `alldifferent` constraint instantiates also $X_4 = 3$. This assignment is part of the dive. Then, $X_2 = 2$ produces the assignment $X_3 = 1$. Suppose now that $X_5 = 7$ is consistent with other problem constraints, while $X_7 = 8$ produces a failure. The dive is therefore: $X_1 = 4, X_2 = 2, X_5 = 7, X_4 = 3, X_3 = 1$.

We want to infer from dives some parameters which measure how good/bad are single variable-value assignments. In particular, as the focus is on the ability of a search heuristic to keep the problem feasible, for a given variable X_i, an assignment $X_i = v$ is considered "better" than an assignment $X_i = u$ if it is part of more feasible solutions.

Our considerations are motivated by two basic conjectures: first, if a variable-value pair $X_i = v$ is part of many feasible solutions, then it is likely to occur in *longer* partial assignments; second, if a variable-value pair $X_i = v$ is part of many feasible solutions, then it is likely to occur in *many* partial assignments. After preliminary experiments we decided to focus on two features based on the above conjectures: the average dive depth (ADD) and the number of occurrences of a given single assignment (OCC). Given a $CSP = (X, D, C)$ and a set of random dives Γ, the *average dive depth* of an assignment $X_i = v_i$ is:

$$ADD(X_i, v_i) = \frac{1}{|\Gamma_{X_i=v_i}|} \sum_{\gamma \in \Gamma_{X_i=v_i}} |\text{dom}(\gamma)|$$

where $\Gamma_{X_i=v_i}$ is the set of dives containing the assignment $X_i = v_i$ and dom(γ) is the set of variables assigned in dive γ. In practice ADD is simply the sample average of the depth of all dives containing a given variable-value pair. With the same notation, the *number of occurrences* as a dive feature is defined as:

$$OCC(X_i, v_i) = |\Gamma_{X_i=v_i}|$$

The following section presents a thorough experimental analysis on the quality of information provided by dives and its relation with the kind of propagation used.

5. Experimental analysis

We performed experiments on 60 Partial Latin Square instances and on reduced-size versions of 9 well known Car Sequencing instances [2], with the primary aim to evaluate the extent of the correlation between dive features and the actual quality of the single variable assignments (density in feasible solutions).

The experimental setting consists of two parts. In the first part, we collected all the useful statistics concerning dive features and actual frequency of variable-value pairs in the feasible solutions. In detail, for each instance we performed the following operations:

A. We computed the actual occurrence frequency of each variable-value pair in the feasible solutions. This evaluation was done via complete enumeration;

B. we collected a number of dives and we used them to extract the ADD and OCC features;

C. we evaluated the variable-value rankings obtained by using ADD and OCC against those obtained by looking at the actual occurrence densities.

The second part of the analysis consists of five groups of experiments performed in order to:

(1) assess the correlation between the dive based scores and the actual densities;

(2) investigate the complementarity of the information provided by different dive features;

(3) estimate the effect of the enforced consistency level on the dive features;

(4) evaluate, on a single-variable basis, the accuracy of a pairwise comparison of the values in the domain, according to the feature provided scores.

(5) assess the effectiveness of the proposed features in guiding actual search runs.

In the following, the adopted models and the instance generation process are detailed for the two target problems. The results attained in each of the five groups of experiments follow.

5.1. Partial Latin Square completion problem

A Partial Latin Square (PLS) of order n is a $n \times n$ matrix where n numbers (e.g., $0, \ldots n-1$) occur once per each row and column; completing a partially filled matrix to a full Latin Square is an NP-complete problem [3]. If the matrix is interpreted as a multiplicative table, then the described structure matches that of Quasi-group in mathematics; hence Latin Square completion problems are sometimes referred to as Quasi-group Completion Problems (QCP).

Randomly generated QCP/PLS instances were shown to exhibit an extremely rapid switch between the under-constrained and the over-constrained region (formally, a phase transition [8]) for a given percentage of pre-filled cells. Instances in the phase transition tend to be the hardest to solve.

[2] Detailed analysis results are available from http://ai.unibo.it/people/MicheleLombardi

For PLS problems of order n, we adopt a classical CP model, based n^2 integer variables $\mathtt{X_{i,j}}$ with domain $\{1, \dots n\}$:

$$\mathtt{alldifferent}([\mathtt{X_{i,1}}, \dots \mathtt{X_{i,n}}]) \qquad \forall i = 1 \dots n \tag{2}$$

$$\mathtt{alldifferent}([\mathtt{X_{1,j}}, \dots \mathtt{X_{n,j}}]) \qquad \forall j = 1 \dots n \tag{3}$$

$$\mathtt{X_{i,j}} \in \{1, \dots n\}$$

Where constraints (2) refer to rows and (3) refer to columns. We generated 60 instances[3], following the method described in [6]; all instances were designed to be in the phase transition, i.e. having 41% of holes for instances of this size. Unless explicitly specified, Generalized Arc Consistency (GAC) was always enforced on the `alldifferent` constraints. The size of the instances must be sufficiently high to make them not too easy, and sufficiently low to allow complete enumeration; after preliminary test we decided to fixed the PLS size (order) to 25.

5.2. Car Sequencing

The Car Sequencing (CS) problem has been originally proposed in [16] and it is now part of the benchmark suite CSPLib[4]. The problem consists in defining an ordering of cars to be produced in an assembly line; the cars differ on the basis of the needed option set to assembly (air-conditioning, sun-roof, and so on). Let k be the number of different configurations, $n_1, \dots, n_k$ are the number of cars for each configuration that need to be produced (with $n = \sum_{1,\dots,k} n_k$); each configuration is described by the set of options that includes. Each option is installed in a different station of the assembly line, however stations have limited capacity and can only process a certain percentage of cars passing along. In particular, each option $o_j \in O$ can be present in at most u_{o_j} cars out of w_{o_j} consecutive cars in order to avoid stalling the assembly line. For example, a given option may be installed in no more than 3 cars out of any consecutive sequence of size 5 of cars.

The task is to find a suitable order of cars such that no stalling of the assembly line occurs and the number of cars produced for each configuration is attained. The problem can be modeled as follows:

$$\mathtt{gcc}([\mathtt{X_1}, \dots \mathtt{X_n}], [\mathtt{n_1}, \dots, \mathtt{n_k}], [\mathtt{n_1}, \dots, \mathtt{n_k}]) \tag{4}$$

$$\mathtt{sequence}([\mathtt{O_{1,j}}, \dots \mathtt{O_{n,j}}], \mathtt{u_{o_j}}, \mathtt{w_{o_j}}) \qquad \forall o_j \in O \tag{5}$$

$$\mathtt{O_{i,j}} = \mathtt{conf_j}[\mathtt{X_i}] \qquad \forall i = 1 \dots n, \forall j = 1 \dots |O| \tag{6}$$

$$\mathtt{O_{i,j}} \in \mathtt{0,1}$$

$$\mathtt{X_i} \in \{\mathtt{1}, \dots \mathtt{k}\}$$

Variable $\mathtt{X_i}$ represents the $i-th$ car of the sequence; variable $\mathtt{O_{i,j}}$ tells whether the $i-th$ car produced has option j or not. Consistency between the variables X and O is maintained by channeling constraints (6): the array element $\mathtt{conf_j}[k]$ is equal to one iff configuration k has option j. Note that variables O are binary and bounding them all directly defines also variables X; in the remainder of the paper, we will consider the set O as the main branching variables.

[3] The target PLS instances are available at http://ai.unibo.it/people/MicheleLombardi

[4] www.csplib.org

Constraint (4) enforces to produce the needed quantity of cars for each configuration; the `sequence` constraints (5) ensure that the assembly line does not stall by forbidding each option to overcome its maximum production capacity. In the experiments, GAC is always enforced on the `gcc` and on the `sequence` constraint.

In order to enumerate all the solutions, we used 9 instances derived by [18][5]. Those instances share the same `sequence` constraints as they respective originals but have lower number of cars to be produced ($n = 30$) and the minimum required number of cars for each configuration is modified. Moreover, unlike their original versions, all the resulting reduced-size instances are feasible. Similarly to the PLS case, the tweaks are motivated by the need to make the target instances non-trivial, but still amenable for complete enumeration.

5.3. Correlation with the actual densities

To evaluate the features proposed, we enumerated all the solutions of the problem instances and computed the densities for each variable-value pair. Formally, the density of a value v_j in the domain of variable X_i is defined as:

$$\rho(X_i, v_j) = \frac{|S_{X_j=v_i}|}{|S|}$$

where S is the set of all feasible solutions of the problem and $S_{X_j=v_i} \subseteq S$ is the set of solutions having $X_j = v_i$. In the following, we also use the notation $\rho(v_j)$ for $\rho(X_i, v_j)$, omitting the target variable when it is clear from the context. We instead write $\sigma(X_i, v_j)$ (or simply $\sigma(v_j)$) to denote the score given to the pair (X_i, v_j) by either *ACC*, *OCC* or *IMP* feature (IMP will be explained later on). Intuitively, we wish the score $\sigma(X_i, v_j)$ to provide as much information as possible about the actual density of the assignment $X_i = v_j$ in feasible solutions.

The correlation between the score and the actual occurrence density was evaluated by defining a set of *error rate indicators* and measuring their value at the root node. The main idea is that the evaluation results for the root should extend to the rest of the search space (of course with some approximation); this comes as a consequence of the fact that the root node is a search node, like any other node in the search tree. The evaluation consists of the following steps:

1. collecting sets of (a fixed number of) dives;
2. computing the error rate indicators for each variable in each instance;
3. extracting statistics for each instance (mean and standard deviation of the error rates *over all the variables* and *over all the sets of dives* from step 1);
4. extracting statistics for all instances (mean and standard deviation for the mean error rates of each instance);

In step 1, we repeated the process with different number of dives, namely 500, 1000 and 2000. For each of such numbers, we collected 30 sets of dives per instance, using different seeds for the random generator; overall, the number of sets of dives for each instance is therefore 3×30. In step 3, mean and

[5]The target CS instances are available at http://ai.unibo.it/people/MicheleLombardi

standard deviation are computed over the pool of data available for each instance, where each variable occurs several times (one per dive set) with different scores.

The error rate indicator computed for the considered dive features are compared with the error rate indicator value corresponding to: (A) the value selection heuristics used in Impact Based Search (*IMP*), and (B) the baseline heuristics consisting in the choice of a value for each variable uniformly at random (*RND*). In particular, Impact Based Search suggests to branch on the variable that has the highest impact (i.e., the amount of reduction of the search space by branching on the variable) and to accord preference to values having the *lowest impact*. Variable and value impacts are extracted in a preliminary step via a fully deterministic process based on domain dichotomization (see [17]), then error based indicators are extracted analogously to the *OCC* and *ADD* dive features. In contrast, error rates for random value selection are computed analytically.

The error rate indicators are discussed extensively in the following; all of them represent *error probabilities* (hence the lower the better) and refer to a specific step of an hypothetic search process. Table 1 and Table 2 respectively report the results of this first evaluation over the PLS and CS instances.

5.3.1. Branching/filtering indicators

The not_max_max indicator assumes a branching process based on the selection of single value for the target variable; in detail, the value with the highest score is chosen, in the hope that it matches the highest actual density in feasible solutions. Ties are broken by choosing a value uniformly at random; in case there is more than one value with highest density, it is sufficient for the chosen value to be among the highest density ones.

In detail, the error rate indicator denotes the probability that a value chosen at random among those with the best *score* does not have the highest *actual density* in the feasible solutions. For the considered variable, let $\sigma(v_i)$ be the score assigned to value v_i by the heuristics; similarly, let $\rho(v_i)$ be the corresponding density in the feasible solutions (number of occurrences/number of feasible solutions). Then this error probability is given by:

$$not\_max\_max = 1 - \frac{|\{v_i \mid \sigma(v_i) = \max_{v_j \in D}(\sigma(v_j)) \wedge \rho(v_i) = \max_{v_j \in D}(\rho(v_j))\}|}{|\{v_i \mid \sigma(v_i) = \max_{v_j \in D}(\sigma(v_j))\}|}$$

where $\max_j(\sigma(v_j))$ is the maximum score assigned by the dive feature to values v_j in the domain of the target variable; similarly, $\max_j(\rho(v_j))$ is the maximum density in feasible solutions.

The not_max_good indicator is similar in spirit; a value v_i for the target variable is chosen as in the previous case, but here we just require v_i to have a "good" density in the feasible solutions. A density is considered "good" if it is greater than or equal to the average density (i.e. the inverse of the variable domain cardinality). Formally, let D be the domain of the target variable, then value v_i has good density iff $\rho(v_i) \geq \frac{1}{|D|}$. Next, the not_max_good indicator can be defined as:

$$not\_max\_good = 1 - \frac{|\{v_i \mid \sigma(v_i) = \max_{v_j \in D}(\sigma(v_j)) \wedge \rho(v_i) \geq \frac{1}{|D|}\}|}{|\{v_i \mid \sigma(v_i) = \max_{v_j \in D}(\sigma(v_j))\}|}$$

In case all values in the variable domain D have the same density, they are all considered good (as no better value could be chosen). The two *not_max* indicators can be useful, for example, if we plan to use dives to design a value selection heuristics.

The not_min_min and not_min_bad indicators are the counterparts of not_max_max and not_max_good; they measure the probability that, for a given variable, a value chosen according to the minimum score has lowest density (resp. bad density). A density is considered "bad" if $\rho(v_i) \leq \frac{1}{|D|}$; hence, if $\rho(v_i) = \frac{1}{|D|}$, the value density is considered *both* good and bad. The *not_min* indicators can be used, for instance, if we plan to use the dives to filter out the least promising values.

The analytical value of all the branching/filtering indicators for the baseline heuristics is obtained by assuming the reference heuristics gives no information at all, so that all values rank equally good and equally bad. In such a situation the error probabilities only depend on the distribution of feasible solutions in the target instance.

5.3.2. DBS based indicators

Indicators $none\_good\_good$ and $none\_bad\_bad$ refer to a Decomposition Based Search process [10], where domains are tentatively classified into a "good" and a "bad" set, before actual assignments begin. Here, a value is classified as good if its score is greater then or equal to the average for the target variable; similarly, a value is classified as bad if its score is lower than or equal to the average. Note that a value can be both good and bad at the same time. In the context of DBS, a critical failure correspond to (A) missing in the "good" set all the values having actual good density or (B) missing in the "bad" set all the values with actual bad density.

The $none\_good\_good$ indicator reports when, for a target variable, a type (A) failure occurs; the average of the indicator over a set of variables is the probability to make such a mistake. The $none\_bad\_bad$ indicator is the counterpart for type (B) failures. Formally, a value v_i is considered to have good score if $\sigma(v_i) \geq \frac{\sum_{v_i \in D} \sigma(v_i)}{|D|}$:

$$none\_good\_good = \begin{cases} 1 & \text{if } \left\{ v_i \mid \sigma(v_i) \geq \frac{\sum_{v_j \in D} \sigma(v_j)}{|D|} \wedge \rho(v_i) \geq \frac{1}{|D|} \right\} = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

$$none\_bad\_bad = \begin{cases} 1 & \text{if } \left\{ v_i \mid \sigma(v_i) \leq \frac{\sum_{v_j \in D} \sigma(v_j)}{|D|} \wedge \rho(v_i) \leq \frac{1}{|D|} \right\} = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Note that in case the heuristic gives no information on a variable, the described DBS process includes all values both in the "good" and in the "bad"; in this situation both *none_good_good* and *none_good_good* are 0 even if the heuristics performance has obviously been bad. In the following, the reader will be notified whenever such situation has non negligible effect on the error rate value.

More in general, the DBS process involves the identification of multiple good/bad values; the general accuracy of such a classification is assessed by the last two indicators. Given a target variable, $good\_bad$ is the probability of a false positive classification, i.e. the probability that a value not having good *density* receives a good *score*. The density of a value v_i is considered "not good" iff $\rho(v_i) < \frac{1}{D}$, hence the concept differs from that of *bad* density ($\rho(v_i) \leq \frac{1}{D}$). Similarly, bad_good is the probability of a

Table 1. Performance indicators and correlation for the dive features on the PLS instances

	dives num	not_max_max	not_max_good	not_min_min	not_min_bad	none_good_good	none_bad_bad	good_bad	bad_good
ADD	500	.62(.03)	.50(.06)	.45(.16)	.20(.02)	.20(.05)	.08(.01)	.53(.06)	.23(.03)
	1000	.58(.03)	.45(.07)	.43(.15)	.18(.02)	.18(.06)	.07(.01)	.50 (.06)	.22(.03)
	2000	.53(.03)	.40(.08)	.41(.15)	.17(.02)	.16(.06)	.06(.01)	.47(.07)	.20(.03)
OCC	500	.42(.05)	.29(.09)	.39(.14)	.15(.02)	.17(.08)	.04(.01)	.32(.09)	.20(.02)
	1000	.41(.05)	.28(.10)	.37(.14)	.13(.02)	.17(.09)	.04(.01)	.31(.09)	.19(.02)
	2000	.40(.05)	.28(.10)	.36(.13)	.12(.02)	.16(.09)	.03(.01)	.31(.09)	.19(.02)
IMP	—	.47(.05)	.34(.09)	.39(.13)	.16(.03)	.21(.09)	.05(.01)	.38(.08)	.22(.03)
RND	—	.74(.01)	.64(.04)	.61(.17)	.36(.04)	.42(.06)	.17(.03)	.74(.11)	.74(.11)

false negative classification. Formally:

$$good\_bad = \frac{\left|\left\{v_i \mid \sigma(v_i) \geq \frac{\sum_{v_j \in D} \sigma(v_j)}{|D|} \wedge \rho(v_i) < \frac{1}{|D|}\right\}\right|}{\left|\left\{v_i \mid \sigma(v_i) \geq \frac{\sum_{v_j \in D} \sigma(v_j)}{|D|}\right\}\right|}$$

$$bad\_good = \frac{\left|\left\{v_i \mid \sigma(v_i) \leq \frac{\sum_{v_j \in D} \sigma(v_j)}{|D|} \wedge \rho(v_i) > \frac{1}{|D|}\right\}\right|}{\left|\left\{v_i \mid \sigma(v_i) \leq \frac{\sum_{v_j \in D} \sigma(v_j)}{|D|}\right\}\right|}$$

The random value of the *none_good_good indicator* (resp. *none_bad_bad indicator*) is the probability to miss all good density (resp. bad density) values by randomly partitioning the domain in two equally sized sets; such probability is analytically computed. The random value for *good_bad* and *bad_good* is the probability of a misprediction, assuming a single value is classified as good/bad with 50% chance.

5.3.3. Discussion of the results

As mentioned, Table 1 reports results for the PLS instances; the first value in each cell is the mean of the indicator, while the standard deviation (over the pool of instances) follows between round brackets. An average dive for the PLS instances counts around 55-65 bound variables.

As a first remark, observe that the selected indicators *do* provide information about the feasibility of variable-value pairs, as both *ADD* and *OCC* dominate *RND*. There is sharp dominance relation between the *OCC*, *IMP* and *ADD* features. In particular, *OCC* beats *IMP* over all the error rate indicators, already for quite small number of dives; this comes quite unexpected and raises a strong interest in using the feature to guide a *value* selection heuristic (although selecting highest density values does not give any performance guarantee in principle); this is investigated in Section 5.7. Conversely, the more complex *ADD* feature provides definitely lower quality feasibility information.

Next, one has to observe values of the standard deviations; since the error rate values over the pool of instance do not have Gaussian distribution, usual statistical relations regarding the amount of value in the interval $[\mu - 2\sigma, \mu + 2\sigma]$ (where μ and σ respectively are the mean and the standard deviation) do not hold. However, the very low values show how all the features tend to report consistent performance over the pool of PLS instances.

The number of collected dives has in principle a strong effect on the quality of any dive based technique. Interestingly, as one can see, the OCC feature is almost no sensible at all to the number of collected dives (at least not with the considered sample sizes). Conversely, the performance of ADD increases more sensibly and the number of dives grows. Note no number of dives is reported for *IMP* and *RND*, as their computation does not involve a dive collection step.

Table 2 reports results for the Car Sequencing problem. As all the branching variables in the adopted problem model are boolean, all the error rate indicators are equivalent and report the same values[6]; hence we preferred to provide instance by instance results (each column is labeled with the corresponding instance name) and report standard deviations over the whole pool of value rankings (i.e. for each set of dives and for each variable). The average dive depth for Car Sequencing instances is around 60-70.

In this setting, the gap between *OCC* and *IMP* feasibility information is even more prominent, while the *ADD* feature provides basically no information (the error rates match those of *RND*). Over different sets of dives, the heuristics tend to make stable guess (either correct or incorrect) in most cases, while they give fluctuating scores to a limited number of variables: this trend is hinted by the higher standard deviations.

The *ADD* feature assigns higher scores to variable-value pairs appearing in long dives. The underlying idea is that longer dives are supposed to contain a bigger proportion of high-density pairs; the reported experiments simply show this is false. A possible explanation is that a small number of "highly infeasible" values is often sufficient to cause a failure, as soon as they are included in the dive: as a consequence, high-density pairs tend to occur in *short* dives rather than in long ones. This also suggests one could save sampling time (and possibly achieve better results) by limiting the dive depth; this investigation is left for future research.

[6]With the exception of *IMP*, reporting lower values for the *none_good_good* and the *none_bad_bad* indicators; however, that is just a consequence of the way those indicators behave when the heuristics is poorly informative (see the note in the description of the *none-...* indicators)

Table 2. Performance indicators and correlation for the dive features on the Car Sequencing instances

	dives	**rp26/82_1**	**rp16/81_1**	**rp41/66_1**	**rp21/90_3**	**rp4/72_3**	**rp6/76_1**	**rp19/71_1**	**rp36/92_1**	**rp10/93_1**
ADD	500	.48(.50)	.52(.50)	.44(.50)	.54(.50)	.53(.50)	.58(.49)	.51(.50)	.37(.48)	.37(.48)
	1000	.48(.50)	.51(.50)	.45(.50)	.55(.50)	.53(.50)	.58(.49)	.52(.50)	.37(.48)	.35(.48)
	2000	.49(.50)	.53(.50)	.44(.50)	.56(.50)	.55(.50)	.59(.49)	.52(.50)	.35(.48)	.34(.47)
OCC	500	.19(.39)	.07(.25)	.15(.35)	.08(.26)	.17(.37)	.09(.29)	.13(.34)	.15(.35)	.10(.29)
	1000	.18(.38)	.07(.25)	.15(.35)	.07(.25)	.16(.36)	.09(.28)	.13(.33)	.15(.35)	.09(.29)
	2000	.17(.38)	.07(.25)	.14(.35)	.07(.25)	.14(.35)	.09(.28)	.12(.33)	.14(.35)	.09(.29)
IMP	—	.27(.34)	.21(.26)	.23(.31)	.17(.25)	.20(.28)	.15(.30)	.18(.31)	.17(.33)	.14(.30)
RND	—	.50(.00)	.49(.06)	.47(.08)	.49(.06)	.49(.06)	.50(.00)	.49(.06)	.48(.10)	.46(.14)

Table 3. Correlation of the dive features on the PLS instances

	not_max_max	not_max_good	not_min_min	not_min_bad	none_good_good	none_bad_bad	good_bad	bad_good
OCC/ADD	.15(.08)	.11(.08)	.26(.09)	.25(.06)	.13(.10)	.23(.13)	.16(.09)	.34(.09)
OCC/IMP	.54(.06)	.49(.08)	.39(.07)	.38(.07)	.44(.10)	.49(.16)	.56(.07)	.58(.07)
ADD/IMP	.03(.07)	.01(.07)	.09(.08)	.10(.05)	.03(.07)	.12(.11)	.04(.07)	.20(.09)

Table 4. Correlation of the dive features on the Car Sequencing instances

	rp26/82_1	rp16/81_1	rp41/66_1	rp21/90_3	rp4/72_3	rp6/76_1	rp19/71_1	rp36/92_1	rp10/93_1
OCC/ADD	-0.19(.06)	-0.08(.06)	-0.14(.06)	-0.04(.04)	-0.16(.07)	-0.28(.03)	-0.26(.06)	-0.05(.06)	-0.13(.06)
OCC/IMP	.64(.02)	.27(.04)	.61(.01)	.35(.02)	.46(.03)	.71(.03)	.65(.03)	.72(.02)	.73(.02)
ADD/IMP	-0.48(.05)	-0.56(.07)	-0.43(.05)	-0.47(.05)	-0.36(.07)	-0.34(.05)	-0.49(.05)	-0.16(.05)	-0.21(.05)

5.4. Complementarity of different dive features

A naturally arising question concerns the possible correlation of the described dive features; in case they are found to provide complementary information, one has the opportunity to exploit such complementarity to increase the detection accuracy. We therefore performed a second set of experiments with the purpose to evaluate the correlation between errors made by pairs of different heuristics. In details:

1. we fixed the number of collected dives to 2000;
2. for each instance, we considered the 30 sets of dives collected in the experiments of Section 5.3;
3. we evaluated the Pearson's r coefficient for the values of all error rate indicators for each instance and set of dives;
4. we averaged over the sets of dives (then on the set of instances, for the PLS problem).

We recall the Pearson's r coefficient (see [19]) ranges in $[-1, 1]$ and evaluates the extent of a linear correlation between two random variables; hence, r close to 1 denotes a strong direct linear correlation and r close to -1 denotes a strong linear inverse correlation (r close to 0 may hint at poor correlation, but technically provides no information, as a non-linear relation may exist).

Table 3 and Table 4 respectively report the correlation analysis results for the PLS and Car Sequencing instances; as in Section 5.3, the Pearson's correlation for each error rate indicator is reported for the PLS (and the standard deviations are computed over the pool of instances), while instance by instance data is shown for the Car Sequencing (and the standard deviations are computed over the sets of dives).

The most relevant result in both cases is the relatively high linear correlation between *OCC* and *IMP*, pointing out that *OCC* and *IMP* tend to make the same mistakes (in terms of feasibility), with *OCC* simply failing less often. On the contrary, no reliable conclusion on *ADD* can be drawn on the PLS, as the r coefficient values are too close to 0. An interesting negative correlation seems to exist between

ADD and *IMP* on the Car Sequencing instances, opening perspectives for the combined use of the two heuristics.

5.5. Effect of the consistency level

As pointed out in Sect.4, dives depend on the actual propagation algorithm used. Therefore, we performed a set of experiments to investigate the effect of the propagation and local consistency on the quality of the feasibility information. As basically all the considered heuristics (included impacts) rely on propagation and diving as a mean to extract information, by weakening the consistency level one may expected a performance drop-off. Table 5 and Table 6 show the results for this evaluation; in particular, we reported the same indicators as in Tables 1 and 2, with the number of collected dives fixed to 2000; Arc Consistency (AC, rather then the strongest consistency level allowed by the solver) is enforced on all the global constraints (`alldifferent`, `gcc`, `sequence`).

On the PLS instances, the performance of all the considered heuristics decreases on average by 5%-10%, with *IMP* and *OCC* being the most sensitive to the achieved consistency level. Conversely, the *ADD* feature is subjected to a less sensible performance drop-off and appears to be more robust w.r.t. propagation. This could be appealing for problems involving complex constraints, for which effective filtering algorithms are not available or not efficient enough.

The situation appears to be completely different on the Car Sequencing instances; here, while *IMP* suffers a limited performance loss when switching to arc consistency, both the dive base heuristics (*OCC* and *ADD*) appear to get no penalty and even report some slightly better results in some cases. This phenomenon is likely connected to the type of branching variables (all binary) and its investigation is left as part of future research.

5.6. Evaluation of the value ranking, on a single variable basis

From a different perspective, the feasibility information can be assessed by evaluating the *ranking* provided for the values in each variable's domain by the dive features and by the actual densities. This method provides an order-based measure of the correlation between the actual occurrence frequency in feasible solutions of variable-value pairs, and their score provided by ADD, OCC or IMP.

In particular, for each variable X_i we computed the number of correctly classified pairs of values

Table 5. Performance indicators for the dive features on PLS, with arc consistency on the `alldifferent` constraints

	dives num	not_max_max	not_max_good	not_min_min	not_min_bad	none_good_good	none_bad_bad	good_bad	bad_good
ADD	2000	.56(.03)	.45(.07)	.45(.16)	.20(.03)	.19(.06)	.07(.01)	.52(.07)	.23(.03)
OCC	2000	.52(.04)	.40(.082)	.43(.16)	.18(.03)	.19(.07)	.05(.01)	.46(.07)	.22(.03)
IMP	—	.52(.04)	.41(.08)	.43(.16)	.18(.03)	.19(.07)	.06(.02)	.47(.07)	.22(.03)

v_j, v_k. A pair v_i, v_j is correctly classified by the ADD (resp. OCC, IMP) feature if:

$$\rho(X_i, v_j) \leq \rho(X_i, v_k) \wedge ADD(X_i, v_j) \leq ADD(X_i, v_k)$$
or
$$\rho(X_i, v_j) > \rho(X_i, v_k) \wedge ADD(X_i, v_j) > ADD(X_i, v_k)$$

where $\rho(X_i, v_j)$ denotes the density of the pair X_i, v_j within feasible solutions. We considered for this of experiments the PLS instances only; for each of them:

1. we counted the number of correctly classified pairs for each variable domain;
2. the number was normalized over the total number of pairs for the variable;
3. the computed value is averaged over all the 30 sets of samples;
4. the resulting mean values are averaged over all the variables in the problem;
5. the resulting mean values are averaged over all the 60 instances in the set.

Table 7 provides the results for this evaluation, conducted on the PLS instances only with 2000, 4000 and 8000 as number of collected dives. The $mean$ and $stdv$ rows report the mean and the standard deviation for the number of correctly classified values (CC) over the set of instances. Here, the IMP heuristic is the one matching more closely the ranking provided by actual densities. Nevertheless, one can see how both ADD and OCC have definitely comparable performance, even outperforming impacts in a minor number of instances (reported in row $> IMP$).

5.7. Effectiveness in guiding search

Finally, we performed tests in order to assess the actual performance one could get by using dive features to guide search. The main issue here is that choosing the values with the highest density in feasible solutions gives in principle no guarantee to obtain the best results; hence (A) the actual relevance of variable-value pair density has to be assessed and (B) the reliability of the error rate indicators in predicting search performance must be evaluated. In other words, the main focus is on how predictable the results are, rather than on the absolute performance. In particular, for the first 10% variables in the problem:

- the branching variable is selected according to a predefined criterion (i.e. maximum impact), to allow a fair comparison;

Table 6. Performance indicators and correlation for the dive features on the Car Sequencing instances

	dives	**rp26/82_1**	**rp16/81_1**	**rp41/66_1**	**rp21/90_3**	**rp4/72_3**	**rp6/76_1**	**rp19/71_1**	**rp36/92_1**	**rp10/93_1**
ADD	2000	.44(.50)	.46(.50)	.54(.50)	.56(.50)	.64(.47)	.58(.49)	.49(.50)	.56(.50)	.50(.50)
OCC	2000	.16(.37)	.05(0.23)	.18(0.38)	.08(.28)	.16(.37)	.10(.31)	.12(.33)	.16(.36)	.11(.31)
IMP	—	.31(.34)	.25(.30)	.24(.29)	.21(.26)	.23(.27)	.21(.34)	.21(.34)	.21(.34)	.21(0.35)

Table 7. Correlation between the ranking provided by dive features and actual densities

		IMP	ADD			OCC		
dives		—	2000	4000	8000	2000	4000	8000
CC	**mean**	.518	.497	.498	.498	.498	.501	.505
	stdv	.024	.018	.015	.013	.026	.023	.023
> IMP		—	8	9	9	13	14	16

- a value is assigned according to the dive scores *collected at the root node*; this choice was forced, as extracting dives at every search node would make an extensive evaluation impractical.

Once at least 10% problem variables are bound, the problem is resolved via a default heuristics (impacts for Car Sequencing, min size domain for the PLS). The main drawback of using scores collected only at the root node is that density information may be compromised along the search: for example the highest density value v_j of a variable X_i may become the lowest one due to a branching decision. Restricting the use of dive related scores to the first 10% variables compensates for the fact that dives are collected at the root node only.

Table 8 and Table 9 show the result of this evaluation, respectively for the PLS and Car Sequencing problems. In detail:

1. each instance was solved once with a default search heuristics, used as a reference (dynamic impacts for Car Sequencing, min size domain for the PLS);
2. each instance was solved once by using actual densities to select values for the first 10% variables;
3. each instance was solved once by using impact (extracted at the root node) to select values for the first 10% variables;
4. each instance was solved 10 times (with different seeds and number of dives fixed to 2000) by using *OCC* to select values for the first 10% variables;

In Table 8 instances are grouped by increasing solution time when performing selection based on densities. For each heuristics we report the mean solution time and fails (and corresponding standard deviations) over the pool of instances. As one can see, selecting value with high occurrence frequency in the feasible solution enables considerable improvements, as pointed out by the lower solution times in the *DENS* columns. *IMP* and *OCC* have similar performance, with *OCC* doing a little better on average; this was indeed predicted by the values of the error rate indicators and provides some experimental support for their relevance.

Table 8. Performance statistics on actual search runs for the PLS instances

DEFAULT		DENS		IMP		OCC	
time	**fails**	**time**	**fails**	**time**	**fails**	**time**	**fails**
45.9(84.1)	117K (219K)	.2(.2)	466(439)	44.1(69.3)	111K(180K)	16.4(29.3)	40K(71K)
19.5(33.5)	53K(98K)	.8(.2)	2K(481)	15.8(16.6)	41K(43K)	30.8(50.6)	82K(136K)
334.2(841.2)	874K(2M)	1.9(.3)	4K(1K)	206.4(561.5)	522K(1M)	219.2(542.1)	559K(1M)
157.5(272.4)	398K(693K)	4.1(1.0)	10K(2K)	69.0(86.7)	170K(210K)	56.5(86.7)	137K(209K)
72.5(121.6)	197K(336K)	15.2(9.7)	40K(26K)	110.4(197.2)	299K(543K)	88.7(131.8)	235K(352K)
139.0(121.9)	365K(328K)	72.9(25.5)	192K(71K)	179.5(313.0)	506K(953K)	99.2(75.9)	256K(191K)

Table 9. Performance statistics on actual search runs for Car Sequencing

	DEFAULT		DENS		IMP		OCC	
inst	**time**	**fails**	**time**	**fails**	**time**	**fails**	**time**	**fails**
rp10/93-1	.58	425	.57	425	.58	425	.57(.01)	421(4)
rp16/81-1	.89	768	.78	579	.78	579	.77(.01)	580(2)
rp19/71-1	.68	541	.61	465	.62	465	.62(.02)	487(22)
rp21/90-3	1.05	1081	.79	721	.78	721	.79(.01)	730(17)
rp26/82-1	.76	622	.67	552	.64	505	1.25(.47)	1298(599)
rp36/92-1	.56	415	.57	451	.57	451	.56(.01)	447(4)
rp41/66-1	.67	540	.59	444	.59	444	.59(.01)	596(15)
rp4/72-3	.62	598	.59	577	.6	577	.61(.02)	474(27)
rp6/76-1	.65	651	.54	511	.54	482	.54(.03)	512(50)

Results for Car Sequencing are shown in Table 9 instance by instance. Note that, to limit the number of solutions and enable enumeration, we had to resort to smaller problems with 30 cars; as a consequence, solution time differences are much less marked. Nevertheless, using exact densities for the first search choices seems to bring some benefit over the default heuristics. Interestingly, impacts at the root node (column *IMP*) provides slightly better results compared to *OCC*, suggesting for this kind of problem other factors (e.g. ability to exploit propagation) play a more important role than matching the highest densities.

6. Conclusion and discussion of future research lines

In the presented paper we have defined a sound theoretical framework for studying the information that can be derived from sampling and diving. In particular, we discussed information concerning the feasibility of variable-value pairs. The extracted information can be used to guide search, or to perform decomposition or to overfilter unpromising values.

A statistical model has been presented that shows that uniform sampling can in principle be informative, but, to be used in practice, it needs huge sample sizes for inferring precise indications. Diving instead can be used for this purpose also in practical cases. We have identified dive features and experimented them on Partial Latin Square and Car Sequencing problems.

We have defined a set of performance indicators to evaluate the effectiveness of each dive features in providing information about the problem variables. The correlation between dive-driven parameters and assignment density in feasible solutions has been measured. Additional experiments have been performed aimed at applying a statistical analysis to measure the complementarity of the features presented and the effect of different consistency levels on the quality of such features. Finally, we performed a last set of experiments by using the identified dive features to actually guide a search process.

A state of the art impact based heuristic was used as a comparison reference throughout the paper. A byproduct of this work is the definition of a statistically sound method for evaluating variable-value selection heuristics.

This work is an attempt to get a better understanding of the search process. Despite the availability of so many methods for the solution of CSPs, the study of the impact on the search efficiency of key search components, such as propagation, branching schema, ability to keep the problem feasible during branching has received poor attention. For this reason, the understanding of CP search behavior by means

of statistics and formal methods is a crucial and very appealing research topic.

The presented work goes is such a direction: it suggests ideas about general evaluation processes, and points out the importance and investigates the limits of maintaining feasibility during search. Many questions remain open and are subject of on-going research. We plan to devise new dive features to be compared with current ones. Second, we are investigating the relations between dive-derived information and backbones and backdoors; on the one hand, to understand which level of accuracy we can reach for such critical variables, and on the other hand to see whether dives can help in their identification. More in general, finding a formal model for some well-known CP search processes is a tough, but very appealing, objective.

Other important issues we are considering are the evaluation of the approach on other problems, the estimation of variables by means of dive-driven information, to detect the most critical ones or those for which we have the highest level of confidence. Finally, we plan to devise techniques to make the diving process more efficient, and to get precise problem information (for evaluation purpose) without resorting to complete enumeration.

References

[1] Beck, J. C.: Solution-Guided Multi-Point Constructive Search for Job Shop Scheduling, *Journal of Artificial Intelligence Research*, **29**, 2007, 49–77.

[2] Boussemart, F., Hemery, F., Lecoutre, C., Sais, L.: Boosting Systematic Search by Weighting Constraints, *Proc. of ECAI*, Ios Pr Inc, 2004, ISBN 1586034529.

[3] Colbourn, C. J.: The Complexity Of Completing Partial Latin Squares, *Discrete Applied Mathematics*, **8**(1), 1984, 25–30.

[4] Gogate, V., Dechter, R.: A New Algorithm For Sampling CSP Solutions Uniformly At Random, in: *Principles and Practice of Constraint Programming - CP 2006* (F. Benhamou, Ed.), vol. 4204 of *Lecture Notes in Computer Science*, Springer-Verlag Berlin Heidelberg, Germany, 2006, 711–715.

[5] Gogate, V., Dechter, R.: Approximate Counting by Sampling the Backtrack-free Search Space, *Proc. of AAAI*, AAAI Press, 2007, ISBN 978-1-57735-323-2.

[6] Gomes, C., Shmoys, D.: Completing Quasigroups Or Latin Squares: A Structured Graph Coloring Problem, *Proc. of the COLOR*, 2002.

[7] Gomes, C. P., Sabharwal, A., Selman, B.: Near-Uniform Sampling Of Combinatorial Spaces Using XOR Constraints, *Advances In Neural Information Processing Systems*, **19**, 2007, 481.

[8] Gomes, C. P., Shmoys, D. B.: Approximations And Randomization To Boost CSP Techniques, *Annals of Operations Research*, **130**(1), 2004, 117–141.

[9] Grimes, D., Wallace, R. J.: Learning To Identify Global Bottlenecks In Constraint Satisfaction Search, *20th International FLAIRS conference*, 2007.

[10] van Hoeve, W., Milano, M.: Decomposition Based Search-A theoretical and experimental evaluation, *Arxiv preprint cs/0407040*, 2004.

[11] Hsu, E. I., Kitching, M., Bacchus, F., Mcllraith, S. A.: Using Expectation Maximization to Find Likely Assignments for Solving CSP'S, *Proc. of AAAI*, 22, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[12] Knuth, D.: Estimating The Efficiency Of Backtrack Programs, *Mathematics of computation*, **29**(129), 1975, 121–136, ISSN 0025-5718.

[13] Le Bras, R., Zanarini, A., Pesant, G.: Efficient Generic Search Heuristics within the EMBP Framework, *Proceedings of the 15th international conference on Principles and practice of constraint programming*, Springer-Verlag, 2009, ISBN 3642042430.

[14] Lombardi, M., Milano, M., Roli, A., Zanarini, A.: Deriving information from sampling and diving, in: *Emergent Perspectives in Artificial Intelligence – AI*IA 2009* (R. Serra, R. Cucchiara, Eds.), vol. 5883 of *Lecture Notes in Computer Science*, Springer-Verlag Berlin Heidelberg, Germany, 2009, 82–91.

[15] Papoulis, A., Pillai, S. U., A, P., SU, P.: *Probability, Random Variables, And Stochastic Processes*, McGraw-Hill New York, 1965.

[16] Parrello, B., Kabat, W., Wos, L.: Job-Shop Scheduling Using Automated Reasoning: A Case Study Of The Car-Sequencing Problem, *Journal of Automated Reasoning*, **2**(1), 1986, 1–42, ISSN 0168-7433.

[17] Refalo, P.: Impact-Based Search Strategies for Constraint Programming, in: *Principles and Practice of Constraint Programming - CP 2004* (M. Wallace, Ed.), vol. 3258 of *Lecture Notes in Computer Science*, Springer-Verlag Berlin Heidelberg, Germany, 2004, 557–571.

[18] Régin, J.-C., Puget, J.-F.: A Filtering Algorithm for Global Sequencing Constraints, in: *Principles and Practice of Constraint Programming - CP 1997* (G. Smolka, Ed.), vol. 1330 of *Lecture Notes in Computer Science*, Springer-Verlag Berlin Heidelberg, Germany, 1997, 32–46.

[19] Rodgers, J. L., Nicewander, W. A.: Thirteen Ways to Look at the Correlation Coefficient, *The American Statistician*, **42**(1), 1988, pp. 59–66, ISSN 00031305.

[20] Ruml, W.: *Adaptive Tree Search*, Ph.D. Thesis, Harvard University Cambridge, Massachusetts, 2002.

[21] Zanarini, A., Pesant, G.: Solution Counting Algorithms For Constraint-Centered Search Heuristics, *Constraints*, **14**(3), 2009, 392–413.