A Study of Adaptive Restarting Strategies for Solving Constraint Satisfaction Problems

Diarmuid Grimes
University College Cork, Ireland, email: d.grimes@4c.ucc.ie

Abstract. In this paper we present a study of four generic strategies for solving constraint satisfaction problems (CSPs) of any kind, be they soluble or insoluble. All four methods combine learning with restarting, two use a fixed cutoff restarting strategy followed by a run to completion, the other two use universal restarting strategies where the cutoff varies from run to run. Learning takes the form of weighting constraints which repeatedly cause failure. This information is then used by a variable ordering heuristic to identify bottleneck variables in search. We evaluate these restarting approaches on scheduling problems and radio link frequency assignment problems (RLFAPs). We show that this form of learning is suited for combining with a restarting approach, and we provide insight into how each approach works.

1 Introduction

In this work we combine two topics which have been the focus of much recent research in the area of CSP solving. The first topic concerns search heuristics which use information from previous search states to guide subsequent search, e.g. the impact-based heuristics of Refalo [1] and the conflict-directed heuristics of Boussemart et al. [2].

The second topic of research concerns the use of restarting strategies combined with an element of randomization for problem solving. Such techniques generally randomize the search heuristics through random tie-breaking, e.g. rapid randomized restarts (RRR) of Gomes et al. [3]. In this case one maintains a degree of confidence in the selections made while still allowing for search diversification. However these methods do not learn from their failed search attempts.

More recently, restarting approaches have been proposed which combine some form of learning with solving. For example, the SAT solver Chaff [4] combines restarting with clause weighting and clause learning. A similar approach has been proposed for combining nogood recording and constraint weighting with restarts in the CSP domain [5]. Refalo also proposed combining restarting with the impact-based heuristics [1]. Gomes refers to these types of approach as "deterministic randomization" [6] where the behaviour of search, although deterministic, is so complex as to appear random.

In this work we concentrate on combining restarting with the weighted-degree heuristic of Boussemart et al. [2]. This heuristic uses constraint weighting to identify variables which are sources of contention in a problem. The heuristic follows the fail-first principle of Haralick and Eliot which states that “to succeed, first try where you are most likely to fail” [7]. By handling these contentious variables higher in the search tree one
increases the likelihood of detecting failures early in the subsequent search. In extreme cases the heuristic can identify insoluble cores in problems [8][9]. It has been shown to be one of the most effective general purpose heuristics [2].

Both restarting and the weighted-degree heuristics are employed to combat the effects of thrashing, where large unpromising parts of the search space are traversed systematically. Restarting allows one to jump out of unpromising search trees. Furthermore, randomized restarting approaches have been proposed as a means to avoid heavy-tail distributions where search costs for a small proportion of problems in the set are several orders of magnitude greater than for the rest of the problems [3].

The weighted-degree heuristic avoids thrashing by moving contentious variables up the ordering. However the heuristic has some drawbacks. The main drawback is that it has the least information available when making its most important choices, i.e. its first few selections. In fact the heuristic has no weight information, other than the degrees of the variables, up until at least one failure has occurred. This drawback was addressed by Grimes and Wallace [9] by combining the heuristic with restarting strategies in two different ways, although one approach used restarting mainly as a sampling process for gathering information. Both of their approaches used a fixed cutoff for a fixed number of runs, followed by a run to completion. We evaluate these two approaches and compare them with the universal restarting strategies of Luby et al. [10] and Walsh [11].

The main contributions of this paper are as follows. We provide an empirical analysis of these four strategies which combine learning with restarting. We show that all four approaches improve on a non-restarting learning strategy. We provide insight as to why this form of learning is suited for combining with restarts, and insight into the behaviour of the different restarting approaches.

The next section provides background. The following section describes the restarting approaches studied, the section thereafter outlines the experimental setup. The following section provides empirical analysis of the different approaches, and the final sections provide a discussion and conclusions of the paper.

2 Background

A constraint satisfaction problem (CSP) can be represented as a tuple of the form \((V, D, C)\) where: \(V = \{V_1, \ldots, V_n\}\) is a set of variables which must be assigned values; \(D = \{D_1, \ldots, D_n\}\) is the set of domains for those variables, consisting of possible values which may be assigned to the variables; \(C = \{C_1, \ldots, C_m\}\) is the set of constraints. These constraints express relations \(\text{Rel}(C_j)\) that state which combinations of values are allowed for the variables in the scope of the constraint, \(\text{Vars}(C_j)\).

Even though all variables will be part of the solution, the order in which variables are selected for instantiation during search can have a large effect on the effort required to find a solution or prove the problem insoluble. Typically a variable is selected according to some heuristic metric obeying the Fail-First principle [7]. One of the most widely used heuristics in CSP solving is that which chooses the variable with minimum ratio of domain size to forward-degree [12] (\(\text{dom/fdeg}\) for short). By choosing the variable with the smallest domain, one is ensuring that if there is to be a failure one will have the fewest alternative values to try. Choosing the variable with the largest forward degree
will generally result in the largest reduction in the underlying search space, through propagation.

The weighted degree heuristic or \textit{wdeg} chooses variables with associated contentious constraints. All constraints are given an initial weight of 1. A constraint’s weight is incremented by 1 each time the constraint causes a failure during search. The weighted degree of a variable is the sum of the weights on constraints between the variable and at least one other uninstantiated variable. The heuristic chooses the variable with largest weighted degree. (Note that this heuristic behaves identically to the max-forward-degree heuristic up until at least one failure has occurred.) In this work we use the \textit{dom/wdeg} variant which chooses the variable with minimum ratio of domain to weighted-degree.

3 Restarting Approaches

We study four different (complete) restarting approaches. All approaches use information learnt, in the form of constraint weights, for guiding subsequent search through the \textit{dom/wdeg} variable ordering heuristic. The first two approaches were proposed by Grimes and Wallace [9], the last two approaches use universal restarting strategies proposed by Luby et al. [10], and Walsh [11] respectively.

The first two approaches, WTDI (WeighTeD Information gathering) and RNDI (RaNDom Information gathering), both have a fixed number of restarts and a fixed cutoff for the first \( R \) runs. On the final run the cutoff is set to infinity and search runs to completion using the \textit{dom/wdeg} heuristic, while continuing to update weights.

The only difference between the two approaches is the variable ordering heuristic used during the first \( R \) runs. WTDI uses the \textit{dom/wdeg} heuristic throughout. Since weights are updated in each run, search is unlikely to revisit a previously explored search tree. This approach behaves like randomized restarting in that it traverses different parts of the search space upon restarting, with the advantage of learning from its failed search attempts. Furthermore, with each successive restart the heuristic has more information from which to make its decisions. Thus each run should improve upon its predecessor. However this assumes that the quality of information learnt is uniform, which is not necessarily the case.

Most problems can either be solved quickly (e.g. problems with multiple solutions), given the right variable ordering, or require extensive search (e.g. insoluble problems with large minimal refutations). WTDI assumes that the problem is the former and performs a number of short search attempts with weights changing the variable ordering from run to run. If the problem is unsolved after many such attempts then it is likely the problem is in the latter category, and so search runs to completion.

The second approach, RNDI (RaNDom Information gathering), selects variables randomly during the first \( R \) runs, and then uses the information from these “random probes” of the search space to guide the early decisions of \textit{dom/wdeg} towards points of most global contention. The aim for each random probe is to provide an outline of areas of contention within the specific search tree the probe is investigating, while avoiding over-weighting local points of contention (hence a low cutoff generally works best). A large number of restarts provides a diverse sample space from which to identify sources of global contention, i.e. variables which are sources of contention across the search
space. The amalgamation of the information from all the probes then indicates which areas of contention are global and to what degree the contention is global.

These approaches may be seen as two extremes, one using an informed search strategy, the other using an uninformed strategy. However if a portfolio of “good” heuristics were used then the weights learnt would be biased. In particular it is of interest to note the minimum and maximum failure depths when assessing such learning strategies. Normally a few variables will be assigned before a failure occurs. Thus the first few variables will receive little if any weight. If one were to use different ‘good’ heuristics for each probe, their top choices would rarely receive any weight.

For RNDI and WTDI, search is restarted after a fixed number of failures (> 1). In their work Grimes and Wallace used a node-based cutoff. Although this fixes the amount of overall search performed, parameters were problem dependent to ensure learning occurred in each run. A failure-based cutoff, on the other hand, ensures the same amount of information is learnt in each run. A cutoff of 30 failures in combination with 100 restarts was used for all problems for both RNDI and WTDI.

The next two restarting approaches have been proposed by Luby et al. [10], and Walsh [11] respectively. The Luby strategy is given by (1,1,2,1,2,4,1,1,2,1,2,4,8,1,...). More formally, for each cutoff $t_i$:

$$t_i = \begin{cases} 2^{k-1} & \text{if } i = 2^k - 1 \\ t_{i-2^{k-1}+1} & \text{if } 2^{k-1} \leq i < 2^k - 1 \end{cases}$$

The geometric strategy is given by $(1, m, m^2, m^3, ....)$ for some constant $m$, where the cutoff increases geometrically. In practice it has been found that a value of $m$ in the range $1 < m < 2$ works best [11]. Following on from the work of Wu and van Beek [13], both Luby and Walsh strategies have a scaled parameter $s$ which is multiplied by each $t_i$ in the sequence. We tested $s = 100$ failures for the geometric approach; and both $s = 100$, and 1000, failures for the Luby approach, as it has been suggested that in practice this approach is often slow to converge [6].

It should be noted that, unlike the other 3 restarting approaches, RNDI is not a true restarting approach in that the restarts are mainly used for information gathering. It is not expected to solve the problem during this phase, but rather to identify the most contentious variables for the weighted degree heuristic on the run to completion.

4 Experimentation

The experiments reported in the next section used JaCoP\(^1\) which is a java constraint programming library. The basic search method used was depth first search with binary branching. Binary branching involves branching on $V_i \neq a$ when the assignment of a value $a$ to a variable $V_i$ was found to be invalid. This has two advantages. Firstly, propagation of $V_i \neq a$ may uncover a failure, and secondly, search may select a different variable to $V_i$ to instantiate even if $V_i$ still has values available in its domain. ($d$-way branching would just select the next value from $V_i$.) Thus binary branching allows for greater variation in the search tree.

\(^1\) http://www.osolpro.com/twiki/bin/view.pl/JaCoP/WebHome
Values were chosen lexically for all approaches. Furthermore all approaches had a maximum of one million nodes in which to find a solution. Due to the number of experiments performed, different machines were used. Thus our results are presented in terms of number of problems solved within the million node cutoff, and in terms of total nodes explored per problem by each approach (including problems that were unsolved within the million node cutoff). We expect there to be a high correlation between runtimes and nodes explored for all approaches. Results for RNDI are averages of 10 experiments, each experiment containing a different random seed.

5 Problem Sets and Results

All problems are taken from the benchmark website of the 2006 CSP Solver Competition\(^2\), and all involve binary constraints. The first set of problems we tested were open shop scheduling problems, the results of which are given in Table 1. These problems were generated by Julien Vion according to specifications given in [14]. There are six sets of 30 problems. The problems are denoted os-taillard-n where n is the number of jobs, the number of operations, and the number of machines.

There are 10 base problems in each set. In order for these problems to be modeled as satisfaction problems (as opposed to optimization), the domains of the operations (variables) have been altered for each problem, placing them into three groupings. For the first group the time windows (domains) are set to the best known makespan for the problem (which may not necessarily be optimal). In the second group the time windows of each operation are increased by 5%, and in the third group the time windows are reduced by 5%. All instances of the first and second group are satisfiable. The second group is generally easiest to satisfy as it has larger domains and thus is likely to have more solutions. Most instances of the third group are unsatisfiable.

All approaches performed well on the first problem set, os-taillard-4. However these problems are relatively easy, e.g. all 20 satisfiable problems were solved during the random probing phase in all ten experiments of RNDI. Once the problems became moderately difficult, dom/fdeg performed poorly, solving at most 10% of the problems.

Crawford and Baker previously described the “early mistake problem” [15] where a large part of the search space, containing no solution, is explored systematically by the algorithm. dom/wdeg acts identically to dom/fdeg up until at least one failure has occurred. Since dom/wdeg always solved more open shop problems than dom/fdeg, the weights learnt enabled search to sometimes recover from / avoid early mistakes encountered by dom/fdeg. The restarting approaches, and in particular the universal restarting approaches, are better equipped to avoid this problem.

Outside of os-taillard-4, RNDI outperformed dom/wdeg on all problem sets, consistently solving more open shop problems. As the problem size increased for these problems, the number solved during the random probing phase decreased. Thus the improvement can be attributed to the weights learnt as opposed to solving the problem during probing. For example, on the largest problem set, RNDI solved three times as many problems as dom/wdeg, even though on average it only solved 1.3 problems dur-
Table 1. Results For Open Shop Scheduling Problems

<table>
<thead>
<tr>
<th></th>
<th>dom - fdeg</th>
<th>dom - wdeg</th>
<th>RNDI</th>
<th>WTDI</th>
<th>Geo WTDI 100WC</th>
<th>Luby WTDI 100WC</th>
<th>Luby WTDI 1000WC</th>
</tr>
</thead>
<tbody>
<tr>
<td>os-taillard-4</td>
<td>25 / 30</td>
<td>30 / 30</td>
<td>30 / 30</td>
<td>30 / 30</td>
<td>30 / 30</td>
<td>30 / 30</td>
<td>30 / 30</td>
</tr>
<tr>
<td># Solved</td>
<td>25 / 30</td>
<td>30 / 30</td>
<td>30 / 30</td>
<td>30 / 30</td>
<td>30 / 30</td>
<td>30 / 30</td>
<td>30 / 30</td>
</tr>
<tr>
<td>Mean Nodes</td>
<td>198K</td>
<td>2K</td>
<td>3K</td>
<td>4K</td>
<td>2K</td>
<td>5K</td>
<td>7K</td>
</tr>
<tr>
<td># Pre Solved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>os-taillard-5</td>
<td>3 / 30</td>
<td>20 / 30</td>
<td>23.4 / 30</td>
<td>25 / 30</td>
<td>26 / 30</td>
<td>24 / 30</td>
<td>23 / 30</td>
</tr>
<tr>
<td># Solved</td>
<td>3 / 30</td>
<td>20 / 30</td>
<td>23.4 / 30</td>
<td>25 / 30</td>
<td>26 / 30</td>
<td>24 / 30</td>
<td>23 / 30</td>
</tr>
<tr>
<td>Mean Nodes</td>
<td>906K</td>
<td>427K</td>
<td>265K</td>
<td>211K</td>
<td>211K</td>
<td>301K</td>
<td>313K</td>
</tr>
<tr>
<td># Pre Solved</td>
<td>-</td>
<td>-</td>
<td>11 / 30</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>os-taillard-7</td>
<td>0 / 30</td>
<td>4 / 30</td>
<td>11.5 / 30</td>
<td>12 / 30</td>
<td>14 / 30</td>
<td>15 / 30</td>
<td>14 / 30</td>
</tr>
<tr>
<td># Solved</td>
<td>0 / 30</td>
<td>4 / 30</td>
<td>11.5 / 30</td>
<td>12 / 30</td>
<td>14 / 30</td>
<td>15 / 30</td>
<td>14 / 30</td>
</tr>
<tr>
<td>Mean Nodes</td>
<td>1,000K</td>
<td>917K</td>
<td>649K</td>
<td>629K</td>
<td>540K</td>
<td>537K</td>
<td>552K</td>
</tr>
<tr>
<td># Pre Solved</td>
<td>-</td>
<td>-</td>
<td>5 / 30</td>
<td>10 / 30</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>os-taillard-10</td>
<td>1 / 30</td>
<td>5 / 30</td>
<td>13.9 / 30</td>
<td>14 / 30</td>
<td>18 / 30</td>
<td>20 / 30</td>
<td>20 / 30</td>
</tr>
<tr>
<td># Solved</td>
<td>1 / 30</td>
<td>5 / 30</td>
<td>13.9 / 30</td>
<td>14 / 30</td>
<td>18 / 30</td>
<td>20 / 30</td>
<td>20 / 30</td>
</tr>
<tr>
<td>Mean Nodes</td>
<td>967K</td>
<td>877K</td>
<td>567K</td>
<td>468K</td>
<td>433K</td>
<td>349K</td>
<td>375K</td>
</tr>
<tr>
<td># Pre Solved</td>
<td>-</td>
<td>-</td>
<td>1.2 / 30</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>os-taillard-15</td>
<td>0 / 30</td>
<td>10 / 30</td>
<td>19.5 / 30</td>
<td>20 / 30</td>
<td>20 / 30</td>
<td>21 / 30</td>
<td>21 / 30</td>
</tr>
<tr>
<td># Solved</td>
<td>0 / 30</td>
<td>10 / 30</td>
<td>19.5 / 30</td>
<td>20 / 30</td>
<td>20 / 30</td>
<td>21 / 30</td>
<td>21 / 30</td>
</tr>
<tr>
<td>Mean Nodes</td>
<td>1,000K</td>
<td>689K</td>
<td>368K</td>
<td>334K</td>
<td>337K</td>
<td>303K</td>
<td>304K</td>
</tr>
<tr>
<td># Pre Solved</td>
<td>-</td>
<td>-</td>
<td>1.3 / 30</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>os-taillard-20</td>
<td>0 / 30</td>
<td>7 / 30</td>
<td>21.2 / 30</td>
<td>22 / 30</td>
<td>22 / 30</td>
<td>22 / 30</td>
<td>22 / 30</td>
</tr>
<tr>
<td># Solved</td>
<td>0 / 30</td>
<td>7 / 30</td>
<td>21.2 / 30</td>
<td>22 / 30</td>
<td>22 / 30</td>
<td>22 / 30</td>
<td>22 / 30</td>
</tr>
<tr>
<td>Mean Nodes</td>
<td>1,000K</td>
<td>808K</td>
<td>328K</td>
<td>270K</td>
<td>274K</td>
<td>270K</td>
<td>271K</td>
</tr>
<tr>
<td># Pre Solved</td>
<td>-</td>
<td>-</td>
<td>1.3 / 30</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The opposite occurred for WTDI, where most solved instances were solved prior to the run to completion for the larger problem sets. The most effective strategies were the different WTDI approaches with the LubyWTDI 100WC solving most problems.

Table 2 gives results for variations of the RLFAP [16] problem scen11. The variations involve the removal of certain frequencies from the variables’ domains as proposed by Bessi`ere [17]. scen11-fn corresponds to scen11 with the n highest frequencies removed from the domains of the variables. All instances are unsatisfiable. Individual results are presented for the 7 problems solved by all approaches (except dom/fdeg as it failed to solve any of these problems).

Neither WTDI nor RNDI solved any of the RLFAP problems prior to the run to completion. However the weights learnt by RNDI during probing clearly improved search over dom/wdeg as it solved the harder instances in much less nodes. RNDI also solved the easier instances extremely quickly on the run to completion, but explored roughly 14,000 nodes in probing, compared to 6,000 nodes explored by WTDI in probing.

Once again the different WTDI approaches performed best. LubyWTDI failed to solve more problems than dom/wdeg, however it was more efficient on the problems solved than dom/wdeg. GeoWTDI solved one more problem than dom/wdeg and was also better in terms of nodes explored, in many cases by an order of magnitude.

Interestingly WTDI solved the most RLFAP problems. Since these problems are insoluble, the proof of insolubility can be quite large for the problems with few frequencies removed. Thus WTDI and RNDI are most suited since these have a long uninter-
Table 2. Results For RLFAP Modified Scen11 Problems

| scen11-f<sub>6</sub> | Mean Nodes | 405,155 | 177,995 | 19,167 | 45,321 | 88,585 | 90,166 |
| scen11-f<sub>7</sub> | Mean Nodes | 248,563 | 109,331 | 14,434 | 20,224 | 37,670 | 39,486 |
| scen11-f<sub>8</sub> | Mean Nodes | 51,611  | 26,723  | 7,450  | 2,666  | 5,931  | 7,709  |
| scen11-f<sub>9</sub> | Mean Nodes | 27,435  | 29,790  | 7,225  | 1,605  | 2,300  | 4,062  |
| scen11-f<sub>10</sub> | Mean Nodes | 1,181   | 14,780  | 6,530  | 410    | 641    | 1,181  |
| scen11-f<sub>11</sub> | Mean Nodes | 1,399   | 14,240  | 6,542  | 1,167  | 1,007  | 1,399  |
| scen11-f<sub>12</sub> | Mean Nodes | 1,399   | 14,414  | 6,542  | 1,167  | 1,007  | 1,399  |

ruptured run. This also explains why neither approach solved any of these problems prior to the run to completion. Contrary to expectation, WTDI produced the better weight profile after probing compared to RNDI, i.e. the better weighted degree ranking of the variables, for solving the problems as it produced the shortest refutations. As one can see from the table, not only did it solve the most problems, but it was also the quickest at solving scen11-f<sub>6</sub> and scen11-f<sub>7</sub>, outperforming dom/wdeg and RNDI by an order of magnitude.

6 Discussion

In the previous section we showed that all restarting approaches outperformed non-restarting approaches on the sample problems. We now provide insight into why these approaches work. Figure 1 displays typical weight profiles generated by RNDI and WTDI on a sample os-taillard-20 problem after 100 restarts with a 30 failure cut-off (the problem was soluble but was unsolved by either approach prior to the run to completion). The slope of the line indicates the level of discrimination between successively ranked variables. As one can see, both approaches offer the most discrimination amongst the top ranked choices. In fact, roughly 50% of the total weight generated is accrued by 10% of the variables. 35% of the variables received no weight increase during the 100 random probes (the original degree of all variables is 38). This means that there were clear differences in the amount of failures accrued by the top ranked variables after the 100 probes. Although the two curves appear similar, different variables were weighted highly by RNDI compared to WTDI.

RNDI clearly identifies certain variables as bottleneck resources in the problem. This can provide valuable feedback for the user as it is a less biased sample of the search space than WTDI, whose search exploration is guided by previous decisions. Furthermore, most of the soluble problems in the os-taillard-20 set were solved by both RNDI and WTDI in 400 nodes on the final run, i.e. virtually backtrack free. However this comes at a cost, there were roughly 38,000 (21,000) nodes explored by RNDI (WTDI resp.) on average during the full probing phase on the os-taillard-20 problems,
compared with roughly 6,500 (6,000 resp.) nodes during the full probing phase on the os-taillard-4 problems.

Grimes and Wallace previously introduced the notion of variable convection for search with a weighted-degree heuristic. This effect describes the rise and fall of variables within the ordering. Variables near the top of the search tree generally receive little weight, when search backtracks these variables fall down the ordering to the point where their constraints are weighted again, they then rise back up the ordering and the cycle repeats. A similar effect occurs in squeaky wheel optimization [18].

In order to assess the impact of this effect, we ran WTDI on the os-taillard-5 set with varying number of restarts, R = 97-104. On a sample problem, variable 20 was selected first after 97, 98, 101 and 103 restarts, while variable 3 was selected first after 99, 100, 102 and 104 restarts. Neither variable received any increase in weight when it was selected first. The number of problems solved within the million node cutoff by WTDI for different R = 97, 98, 99, 100 and 101 was 25, 23, 25, 25, 24 respectively. This illustrates the magnitude of the variation in search performance as the variable ordering changes due to the convection effect. One extra run (with a 30 failure cutoff) can lead to (not) solving two extra problems. This means there can be a certain amount of luck with regard to the quality of the initial variable ordering on the run to completion using WTDI.

However this effect also illustrates why the weighted-degree heuristic is ideally suited to a restarting strategy such as GeoWTDI or LubyWTDI. When using a restarting strategy one would like to minimise the possibility that search will explore the same search tree on successive runs. Since the weight profile never stabilises (unless the problem contains an insoluble core), the ordering is constantly evolving from run to run, with different variables rising up and falling back down.
In the geometric approach, the ordering will be biased to the sources of contention in the previous run, since this had the most failures. However, in the case of the Luby approach, one often has a long run followed by a number of short runs. In this case the same search tree may be explored in the short runs, as the weights from the long run may subsume those learnt on the short runs. In order to test this we outputted the weighted degrees of the variables after different numbers of restarts for both Luby and geometric approaches, on a sample os-taillard-20-95 problem which was unsolved by any approach (this allowed us to test the case when the Luby strategy had a very long run followed by a number of short runs).

In the case of the Luby strategy we generated weight profiles after 255, 256, 257 and 258 restarts. The cutoff for the 255th restart was 12800 failures while the cutoff for the following three restarts was 100, 100, 200 respectively. The ranking by weighted degree of the top 20 variables changed quite dramatically between the start of the 255th restart and the 256th restart (as expected after the large cutoff). However there was no change in the ranking of the variables between the 256th restart and the 257th restart, and only a slight change (a swapping of two variables ranked 13th and 14th) between restart 256 and 257. For the geometric approach we ranked the variables by their weighted degree after the 14th and 15th restart. Only two of the top 20 variables (ranked 19th and 20th) had the same rank. This confirms our intuition that the geometric approach is generally best suited to the weighted degree heuristic as it is less likely to repeatedly explore the same search tree.

Interestingly only 20 variables (out of 400) received any weight on this problem when using any of the WTDI approaches. On closer inspection these variables form a clique as they are all operations in the one job. The earliest starting time of each operation in this clique if it was chosen last (given by the sum of the durations of the other tasks) was always greater than the operations maximum domain value, thus these variables form an insoluble core and this is identified by their constraint weights. This is a clear advantage over a non-learning randomized restarting approach which would not be able to identify the insoluble core. It was also of interest to note that the WTDI approaches “zoned in” on this subset of variables and repeatedly weighted them. Although RNDI did weight variables in the clique highly for this problem, it also weighted other variables with the result that search “jumped around” on the run to completion instead of focussing on the clique.

We found a similar effect on a sample RLFAP problem. WTDI only weighted 22 of the 680 variables. These 22 variables were connected and, taken together with their constraints, were sufficient to prove the problem insoluble. RNDI weighted over 600 of the variables, albeit weighting some much higher than others. This weighting of variables in disparate parts of the constraint graph by RNDI is a possible explanation for the difference in results between these two approaches on the RLFAP problems, as jumping around in the constraint graph is detrimental to the build up of contention.

7 CONCLUSION

In this work we provided a study of four restarting approaches in combination with the weighted-degree heuristic. We illustrated the benefits of combining restarting with
learning on real world problems, with all restarting approaches outperforming the non-
restarting strategies. We showed, through the notion of variable convection, why this
heuristic is suited to restarting strategies. We also provided insight as to which restart-
ing approach is generally most suited to the weighted-degree heuristic (the geometric
approach), and insight into the advantages and disadvantages of each approach.

References

2. Bousssemart, F., Hemery, F., Lecoutre, C., Sais, L.: Boosting systematic search by weighting
(2007) 131–136
8. Hemery, F., Lecoutre, C., Sais, L., Bousssemart, F.: Extracting MUCs from constraint net-
12. Bessi`ere, C., R´egin, J.C.: Mac and combined heuristics: Two reasons to forsake fc (and cbj?)
1118, (1996) 61–75
13. Wu, H., van Beek, P.: On universal restart strategies for backtracking search. In Bessi`ere,
681–695
Research 64 (1993) 278–285
15. Crawford, J.M., Baker, A.B.: Experimental results on the application of satisfiability algo-
16. Cabon, B., de Givry, S., Lobjois, L., Schiex, T., Warners, J.P.: Radio link frequency assign-
17. Bessi`ere, C., Chmeiss, A., Sais, L.: Neighborhood-based variable ordering heuristics for
the constraint satisfaction problem. In: Principles and Practice of Constraint Programming-
CP’01. LNCS No. 2239, (2001) 565–569