Toward Understanding Solution-Guided Multi-Point Constructive Search for CSPs

J. Christopher Beck & Ivan Heckman

Department of Mechanical & Industrial Engineering
University of Toronto
{jcb,iheckman}@mie.utoronto.ca

Abstract. Solution-Guided Multi-Point Constructive Search (SGMPCS) is a complete, constructive search technique that has been shown to out-perform standard constructive search techniques on a number of constraint optimization and constraint satisfaction problems. In this paper, we perform a case study of the application of SGMPCS to a constraint satisfaction model of the multi-dimensional knapsack problem. We show that SGMPCS performs poorly. We then develop a descriptive model of its performance using fitness-distance analysis. It is demonstrated that SGMPCS search performance is partially dependent upon the correlation between the heuristic evaluation of the guiding solutions and their distance to the nearest satisfying solution. This is the first work to develop a descriptive model of SGMPCS search behaviour. The descriptive model points to a clear direction in improving the performance of constructive search for constraint satisfaction problems: the development of heuristic evaluations for partial solutions.

1 Introduction

An important line of research over the past 15 years in combinatorial optimization has been the empirical study of average algorithm behaviour. For example, there has been significant study of phase transition phenomena [7, 17], work on heavy-tailed distributions [8, 12], and detailed models developed for tabu search for job shop scheduling [19]. In this paper, we build on this work to begin to develop an understanding of the search behaviour of a recently proposed constructive search technique, Solution-Guided Multi-Point Constructive Search (SGMPCS) [3].

We examine the performance of SGMPCS on a set of benchmark instances of a constraint satisfaction version of the multi-dimensional knapsack problem. We show that both randomized restart and SGMPCS perform poorly on these instances when compared to chronological backtracking. The core of the paper is the investigation of the conjecture that SGMPCS performance, unlike that of randomized restart and chronological backtracking, is partially affected by the quality of the heuristic that is used to select the guiding partial solutions. When we artificially control the quality of the heuristic evaluation, we observe substantial performance differences. We then investigate two new heuristics. The better heuristic results in significant gain in search performance and, more importantly, the observed performance differences among the three heuristics are consistent with the descriptive model. Approximately 44% of the variation in search performance can be accounted for by the quality of the heuristic.

This is the first work which demonstrates that SGMPCS exploits the heuristic evaluation of its guiding solutions. Given that standard constructive search techniques do not exploit this information, we believe that SGMPCS embodies a promising direction for improving constructive search performance.

In the next section, we present SGMPCS, briefly discuss the literature on empirical models of search behaviour, and introduce the multi-dimensional knapsack problem. In Section 3, we present and discuss the initial empirical studies, demonstrating the poor performance of SGMPCS. Section 4 develops our descriptive model of SGMPCS performance. Section 4.4 proposes two new heuristic evaluation functions and empirically evaluates them. We discuss the implications and limitations of our study in Section 5.

2 Background

In this section, we present the Solution-Guided Multi-Point Constructive Search algorithm, previous work on building descriptive models for search performance, and introduce the multi-dimensional knapsack problem.

2.1 Solution-Guided Multi-Point Constructive Search

Solution-Guided Multi-Point Constructive Search (SGMPCS) [3, 1] is a constructive search technique originally proposed for optimization problems. For clarity, we present the basic approach in the optimization context first before discussing the changes necessary for constraint satisfaction problems.

SGMPCS for Optimization The primary novelty of SGMPCS is that it is guided by sub-optimal solutions that it has found earlier in the search. As with randomized restart techniques [8], the overall search consists of a series of tree searches limited by a computational resource bound. When the resource bound is reached, search restarts and may be guided by an *elite* solution. An elite solution is a high-quality, sub-optimal solution found earlier in the search.

Pseudocode for SGMPCS is shown in Algorithm 1. The algorithm initializes a set, e, of elite solutions and then enters a while-loop. In each iteration, with probability p, search is started from an empty solution (line 5) or from a randomly selected elite solution (line 10). In the former case, if the best solution found during the search, s, is better than the worst elite solution, s replaces the worst elite solution. In the latter case, s replaces the starting elite solution, r, if s is better than r. Each individual search is limited by a fail bound: a maximum number of fails that can be incurred. The entire process ends when the problem is solved, proved insoluble within one of the iterations, or when some overall bound on the computational resources (e.g., CPU time, number of fails) is reached.

Elite Solution Initialization The elite solutions can be initialized by any search technique. For each problem, we use independent runs of standard chronological backtracking with a random variable and value ordering. The search effort is limited by a maximum number of fails for each run.

Algorithm 1: SGMPCS: Solution-Guided Multi-Point Constructive Search

```
    SGMPCS():
 1  initialize elite solution set e
 2  while not solved and termination criteria unmet do
 3      if rand[0, 1) < p then
 4          set fail bound, b
 5          s := search(∅, b)
 6          if s is better than worst(e) then
 7              replace worst(e) with s
        else
 8          r := randomly chosen element of e
 9          set fail bound, b
10          s := search(r, b)
11          if s is better than r then
12              replace r with s
```

Bounding the Search Each individual search is bounded by an evolving fail bound: a single search (lines 5 and 10) will terminate, returning the best solution encountered, after it has failed the corresponding number of times.

Searching from an Empty Solution With some probability, p, search is started from an empty solution (line 5). Searching from an empty solution simply means using any standard constructive search with a randomized heuristic and a bound on the number of fails. In our experiments, the search from an empty solution uses the same search techniques used to initialize elite solutions.

Searching from an Elite Solution To search from an elite solution, we create a search tree using any variable ordering heuristic and specifying that the value assigned to a variable is the one in the elite solution, provided it is still in the domain of the variable. Otherwise, any other value ordering heuristic can be used to choose a value. Formally, given a constraint satisfaction problem (CSP) with n variables, a solution, s, is a set of variable assignments, $\{\langle V_1 = x_1\rangle, \langle V_2 = x_2\rangle, \ldots, \langle V_m = x_m\rangle\}, m \leq n$. When $m = n$, the solution is complete, but possibly infeasible; when $m < n$, s is a partial solution. A search tree is created by asserting a series of choice points of the form: $\langle V_i = x\rangle \vee \langle V_i \neq x\rangle$ where V_i is a variable and x the value that is assigned to V_i. The variable ordering heuristic has complete freedom to choose a variable, V_i, to be assigned. If $\langle V_i = x_i\rangle \in s$ and $x_i \in dom(V_i)$, the choice point is made with $x = x_i$. Otherwise any value ordering heuristic can be used to choose $x \in dom(V_i)$. The only difference between starting search from an empty solution and from an elite solution is that the latter uses the assignments of the elite solution as a value ordering heuristic.

Adapting SGMPCS for CSPs To apply SGMPCS to constraint satisfaction problems, it is necessary to define what an elite solution is and how one is evaluated. The elite solutions are used as value ordering heuristics and the evaluation of a solution determines

whether it will be used to guide subsequent search. Therefore, the evaluation of a potential elite solution is a *heuristic evaluation*. Since the only purpose of an elite solution is to provide value-ordering guidance, we want our evaluation function to choose elite solutions likely to guide the search to a satisfying assignment.[1] We experiment with three different heuristic evaluation functions as described in Sections 3.1 and 4.4.

We define elite solutions as dead-ends: either an assignment to a proper subset of the variables that results in a domain wipe-out or a complete assignment for which one or more constraints is broken. The solver may visit a complete assignment of variables that is not a solution to the problem. Consider a situation where $n-2$ variables have been assigned in lexicographical order and v_{n-1} and v_n both have non-empty, non-unary domains. The assignment of v_{n-1} may trigger the reduction of the domain of v_n to a singleton, followed by its immediate assignment, while there are still constraints in the propagation queue. If one of these other constraints is not satisfied by the now-complete assignment, we have a complete assignment that fails to satisfy all constraints. The rating of a dead-end is done, as noted above, with a heuristic evaluation function.

We identify the dead-ends that are candidates for the elite set by modifying the behaviour of the solver to keep track of the highest rated dead-end found during a single search (i.e., during the calls at 5 and 10 of Algorithm 1). Whenever a dead-end is found, it is compared to the stored dead-end and the latter is replaced if the former is more highly rated. At the end of a single search that has not found a satisfying solution, the best dead-end is returned and is considered for insertion into the elite set.

An alternative approach is to adopt a soft constraint framework where each potential elite solution is a complete assignment that breaks one or more constraints and the evaluation is an aggregation of the cost of the broken constraints. This is an interesting area for future work, but we do not consider it here for a number of reasons.

- We are motivated by simplicity and the desire to modify the behaviour of standard (crisp) constraint solvers as little as possible.
- A soft constraint approach cannot fully exploit the strong constraint propagation techniques that are one of the core reasons for the success of CP.
- It is unclear *a priori* which cost models for various global constraints [4, 18] is appropriate for the purposes of providing a heuristic evaluation.

We return to the question of a soft constraint model in Section 5.2.

2.2 Descriptive Models of Algorithm Behaviour

A descriptive model of algorithm behaviour is a tool used to understand why an algorithm performs as it does on a particular class or instance of a problem. There has been considerable work over the past 15 years in developing models of problem hardness [7, 17] as well as work that has focused more directly on modeling the behaviour of specific algorithms or algorithm styles. The work on heavy-tailed phenomenon [8, 12]

[1] This is true for optimization contexts as well. However, the existence of a cost function obscures the fact that guiding the search with sub-optimal solutions is only useful if such guidance is likely to lead the search to lower cost solutions.

models the dynamic behaviour of constructive search algorithms while local search has been addressed in a number of models–see [10] for a detailed overview.

In this paper, we develop a static cost model with the goal of correlating problem instance features to algorithm performance. Our primary interest is to understand why SGMPCS outperforms or, as we will see below, fails to outperform, other constructive search techniques. The approach we adopt is *fitness-distance analysis* [10] an *a posteriori* approach traditionally applied to local search algorithms. Local search algorithms move through the search space based on an evaluation of the quality of (suboptimal) "solutions" in the neighborhood of the current solution. Neighboring solutions are evaluated and, typically, the lowest cost solution is selected to be the next solution. In fitness-distance analysis, the quality of a solution (i.e., its *fitness*) is compared against its distance to the nearest optimal solution. Distance is measured as the minimum number of steps it would take to move from the solution in question to the nearest optimal solution. In problem instances where the search space and neighborhood function induce a high *fitness-distance correlation* (FDC), the standard behaviour of moving to a solution with higher fitness will also tend to move the search closer to an optimal solution.

Standard constructive search techniques such as chronological backtracking, limited discrepancy search, and randomized restart do not exploit the fitness of sub-optimal solutions that are found during search. Even when there is a notion of sub-optimal solution, as in optimization problems, these techniques do not attempt to search in the "neighborhood" of high quality solutions. There are, however, some algorithms that are based on constructive search such as ant colony optimization [5] and adaptive probing [16] that have been shown to be sensitive to FDC on optimization problems [2].

We test the hypothesis that SGMPCS is sensitive to fitness-distance correlation and that, therefore, its search performance can be partially understood by the FDC of a problem instance.

2.3 The Multi-Dimensional Knapsack Problem

Given n objects and a knapsack with m dimensions such that each dimension has capacity, $c_1, \ldots, c_m$, a *multi-dimensional knapsack problem* requires the selection of a subset of the n objects such that the profit, $P = \sum_{i=1}^{n} x_i p_i$, is maximized and the m dimension constraints, $\sum_{i=1}^{n} x_i r_{ij} \leq c_j$ for $j = 1, \ldots, m$, are respected. Each object, i, has a individual profit, p_i, a size for each dimension, r_{ij}, and a binary decision variable, x_i, specifying whether the object is included in the knapsack ($x_i = 1$) or not ($x_i = 0$).

There has been significant work on such problems in the operations research and artificial intelligence literature [13, 6]. Our purpose, is not to compete with these approaches. Rather, our purpose is to develop an understanding of the behaviour of SGMPCS. We selected the multi-dimensional knapsack problem because previous work has indicated that SGMPCS performs particularly poorly on such problems and we want to understand why [9].

Because we are solving a constraint satisfaction problem, we adopt the approach of [15] and pose the problem as a satisfaction problem by constraining P to be equal to the known optimal value, P^*. In addition to the constraints defined above, we therefore add $P = P^*$.

3 Initial Experiment

In this section, we present the details and results of our initial experiments.

3.1 Experimental Details

We compare three search techniques: chronological backtracking (*chron*), randomized restart (*restart*) [8], and SGMPCS. In all algorithms, the variable ordering is random. The value ordering for each algorithm, when not being guided by an elite solution, is also random. Any restart-based technique needs some randomization. The use of purely random variable and value ordering serves to simplify the experimental set-up.

Restart follows the same fail sequence as SGMPCS (see below) and initializes and maintains a set of elite solutions. However, it always searches from an empty solution (i.e., it is equivalent of SGMPCS with $p = 1$). Therefore, it has a small run time overhead to maintain the elite set as compared with standard randomized restart.

All algorithms were implemented in ILOG Solver 6.3 and run on a 2GHz Dual Core AMD Opteron 270 with 2GB RAM running Red Hat Enterprise Linux 4.

Parameter Values for SGMPCS Previous work has examined the impact of different parameter settings [3, 1]. Here, we are interested in SGMPCS performance in general, and, therefore, adopt the following parameters for all experiments.

- Probability of searching from an empty solution: $p = 0.5$.
- Elite set size: $|e| = 8$.
- Backtrack method: chronological. For a single search, we have a choice as to how the tree search should be performed at lines 5 and 10.
- Fail sequence: Luby [14]. The fail sequence sets the number of fails allowed for each tree search. The Luby sequence corresponds to the optimal sequence when there is no knowledge about the solution distribution: 1,1,2,1,1,2,4,1,1,2,1,1,2,4,8, ... Following [11], we multiply each limit by a constant, in our case 32.
- Number of initial solutions: 20. At line 1 we generate 20 partial solutions and then choose the $|e|$ best to form the initial elite set.
- Initialization fail bound: 1. The effort spent in finding a good initial solution is controlled by the fail bound on the search for each initial solution. We simply stop at the first dead-end found.

These parameters were chosen based on previous work and some preliminary experiments that showed little performance variation for SGMPCS for different settings on multi-dimensional knapsack problems [9].

Problem Instances Two sets of six problems from the operations research library[2] are used. The instances range from 15 to 50 variables and 2 to 30 dimensions.

For each problem instance, results are averaged over 1000 independent runs with different random seeds and a limit of 10,000,000 fails per run. For each run of each problem instance, we search for a satisfying solution.

[2] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html

Heuristic Evaluation for SGMPCS Following the idea of trying simple approaches before more complex ones, our initial heuristic evaluation is the number of unassigned variables. Recall that our elite solution candidates are dead-ends (Section 2) that either have one or more variables with an empty domain or break a constraint. When the solver encounters a dead-end, we simply count the number of unassigned variables and use that as the heuristic evaluation: the fewer unassigned variables, the better the dead-end. We make no attempt at a dead-end to assign any of the unassigned variables that have non-empty domains. We refer to this heuristic evaluation as H_1. This heuristic has shown strong performance on quasigroup-with-holes completion problems [1].

3.2 Results

Table 1 compares the performance of chronological backtracking, randomized restart, and SGMPCS as defined above. Both SGMPCS and randomized restart perform poorly when compared to chronological backtracking. There does not seem to be a large difference between the performance of SGMPCS and randomized restart.

	chron			restart			SGMPCS-H_1		
	%sol	fails	time	%sol	fails	time	%sol	fails	time
mknap1-0	100	**1**	0.0	100	2	0.0	100	3	0.0
mknap1-2	100	**26**	0.0	100	42	0.0	100	41	0.0
mknap1-3	100	**523**	0.0	100	1062	0.0	100	924	0.0
mknap1-4	100	**15123**	**0.4**	100	54635	1.5	100	44260	1.2
mknap1-5	100	**3271555**	**67.2**	54.5	6885489	167.2	70.8	5573573	137.0
mknap1-6	0.2	9990291	**279.9**	0.0	10000000	337.2	0.8	**9958245**	340.9
mknap2-PB1	100	**15223**	0.3	100	42651	0.8	100	28770	0.6
mknap2-PB2	100	**3088092**	**54.1**	80.3	4970049	102.0	88.1	3741187	77.8
mknap2-PB4	100	**10167**	0.1	100	38474	0.5	100	28406	0.4
mknap2-PB5	100	**7011**	0.1	100	16178	0.4	100	15077	0.3
mknap2-PB6	100	**16050**	**1.9**	100	28964	3.8	100	25954	3.4
mknap2-PB7	100	**1472499**	**138.7**	76.0	5374900	551.4	85.9	4113704	423.6

Table 1. Comparison of multi-dimensional knapsack results for chronological backtracking (*chron*), randomized restart (*restart*) and SGMPCS using the H_1 heuristic evaluation function.

Given previous results that showed SGMPCS out-performing randomized restart and chronological backtracking on optimization problems [3] and quasigroup-with-holes constraint satisfaction problems [1], the multi-dimensional knapsack results motivate the balance of this paper.

4 Building a Descriptive Model

Beck [3] speculates that three, non-mutually exclusive, factors may have an impact on the performance of SGMPCS: the exploitation of heavy-tails, the impact of revisiting elite solutions, and the use of multiple elite solutions to diversify the search. Here we

focus on developing a descriptive model based on the second factor. The intuition behind this factor is that each time a good solution is revisited with a different variable ordering a different set of potential solutions (i.e., a different "neighbourhood") will be visited upon backtracking. If good solutions tend to be near other good solutions in the search space, revisiting a solution is likely to result in finding another good solution.

In this section, we develop a descriptive model of SGMPCS performance based on the fitness-distance correlation. We first define the measure of distance used and then present a deeper analysis of the SGMPCS results in the above table. We then build on the methodology of Beck & Watson [2], to create an artificial heuristic evaluation function that allows us to completely control the fitness-distance correlation of the problem instances. Experiments with this artificial heuristic demonstrate a strong interaction between FDC and search performance. Finally, we develop two new heuristic evaluation functions and examine their performance.

4.1 A Measure of Distance

A complete solution to a multi-dimensional knapsack problem can be represented by a binary vector $(x_1, ..., x_n)$ of the decision variables. The representation lends itself to using the Hamming distance as a measure of the distance between two (complete) assignments. This is the standard definition in fitness-distance analysis.[3]

Our elite solutions are dead-ends and so may not be complete assignments. Therefore, we must adapt the Hamming distance to account for unassigned variables. A given dead-end with m assigned variables, $m < n$, represents a set of 2^{n-m} points in the search space with varying distances from the nearest satisfying solution. If we assume a single satisfying solution to a problem instance (see below), then the distribution of distances for the sub-vector of unassigned variables follows a binomial distribution with a minimum sub-distance of 0 and maximum sub-distance of $n-m$. The mean of this distribution is $\frac{n-m}{2}$. We therefore calculate the distance from a dead-end to the satisfying solution as the mean distance of the points represented by the dead-end: the Hamming distance for the assigned variables plus one-half the number of unassigned variables. More formally, for a given elite solution candidate $S = (x_1, ..., x_m)$ and a satisfying solution $S^* = (x_1^*, ..., x_n^*), m \leq n$, the distance is calculated as follows:

$$D(S, S^*) = \sum_{1 \leq i \leq m} |x_i - x_i^*| + \frac{n-m}{2} \tag{1}$$

The normalized distance is $ND(S, S^*) = \frac{D(S,S^*)}{n}$.

4.2 Analysis of the Initial Experiments

Traces of SGMPCS-H_1 runs show that early in the search all the elite solutions have a heuristic evaluation of 0: all the variables are assigned but the solution does not sat-

[3] SGMPCS does not move in the search space with the freedom of local search as it is constrained by a search tree. A different definition of distance that takes into account the search tree may be more appropriate. We leave the investigation of such a distance function for future work.

isfy all constraints. The uniformity of the heuristic evaluation suggests that our simple heuristic evaluation is too coarse to provide useful guidance.

To quantify this observation, we calculate the heuristic evaluation and distance of each elite solution encountered during the search. In order to do this, we must first find all satisfying solution to each instance. We did this using a small modification to the chronological backtracking algorithm. To our surprise, each instance has a single satisfying solution, justifying our definition of D above.

Figure 1 presents plots of the distance vs. the fitness for two of the problem instances. The plots for the other problem instances are almost identical. It is clear, that the heuristic evaluation provides almost no real heuristic information. These data were gathered by instrumenting the SGMPCS solver to record the fitness and distance from the known satisfying solution of each new entry to the elite set.

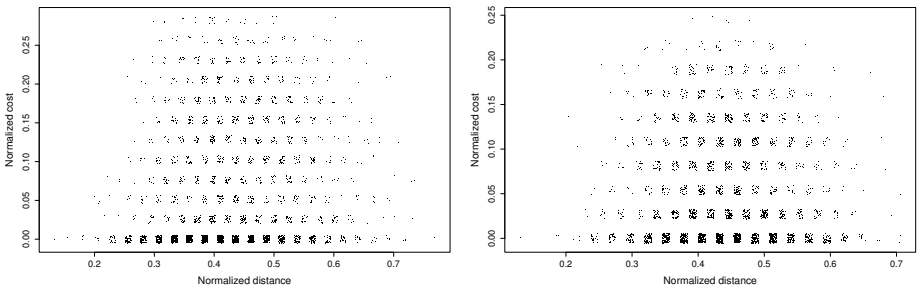

Fig. 1. Plots of the heuristic evaluation (fitness) of each elite solution vs. its normalized distance from the unique satisfying solution for two of the multi-dimensional knapsack problem instances: Left: mknap1-5; Right: mknap2-PB7. A small noise component is added to the fitness and distance for purposes of visibility in the plot–this noise is not present in the data.

4.3 Manipulating the Fitness-Distance Correlation

Figure 1 is consistent with our conjecture that fitness-distance correlation may have a role in a descriptive model of SGMPCS performance. It provides, however, rather weak support: the absence of an FDC accompanies poor performance. A stronger test of the conjecture is to directly manipulate the FDC and observe the performance of SGMPCS. To do this, we adopt the technique introduced in [2] to artificially set the heuristic evaluation based on knowledge of the distance to the satisfying solution.

Let $D(S, S^*)$ be defined as in Equation (1). We define the heuristic evaluation of the satisfying solution, S^*, to be $h(S^*) = 0$. We set the heuristic evaluation, $h_{FDC+}(S)$, of an elite solution S under perfect FDC equal to $D(S, S^*)$. Similarly, we set the heuristic evaluation $h_{FDC-}(S)$ of an elite solution with perfect negative FDC to be $(n - D(S, S^*))$. To generate instances with intermediate FDC, we interpolate between these two extremes as follows:

$$h(S) = \begin{cases} 0 \text{ if } S = S^* \\ \lceil \alpha \times h_{FDC+}(S) + (1-\alpha) \times RAND(S) \rceil \text{ if } S \neq S^* \wedge \beta = 0 \\ \lceil \alpha \times h_{FDC-}(S) + (1-\alpha) \times RAND(S) \rceil \text{ if } S \neq S^* \wedge \beta = 1 \end{cases} \quad (2)$$

where $\alpha \in [0,1]$, $\beta \in \{0,1\}$, and $RAND(S) \in [0,n]$; the latter value is uniformly generated from the interval, using the bit vector S as the random seed. The random component is added to achieve more realism in our model, while still manipulating the FDC. Clearly, when $\alpha = 0$, the heuristic evaluation is purely random. While α determines the strength of the FDC, β is a two-valued parameter governing its direction: $\beta = 0$ and $\beta = 1$ induce positive and negative FDC, respectively.

The only difference with our initial experiments is that the heuristic evaluation is changed to Equation (2). For a single instance and each pair of values for α and β, we solve the instance 1000 times with different random seeds. Following Watson [19] we compare FDC against the log of search cost, in our case, the log of the number of fails to find a satisfying solution. Since our problems are of various sizes, the log of the mean number of fails of instance p with $\alpha = a, \beta = b$, $\bar{F}_{p,a,b}$, is normalized with the log of the search cost of *chron* on the same problem (C_p) as follows:

$$N_{p,a,b} = \frac{log(\bar{F}_{p,a,b}) - log(C_p)}{log(C_p)}$$

For each problem and setting of α and β, FDC values are measured by collecting every unique elite solution over the 1000 iterations and taking the correlation between the evaluation function for each entry and its distance to the one known satisfying solution as defined in Equation 1.

Figure 2 shows that the manipulation of the FDC has a significant impact on the search performance of SGMPCS. The graph does not contain results for mknap1-0 and mknap1-2. As shown in Table 1, these are easily solved during the initialization phase of SGMPCS and so display no correlation with FDC. There is considerable noise for high negative values of FDC due to the fact that SGMPCS could not find a solution on a number of problem instances with high negative FDC, within the fail limit.

4.4 Toward Better Heuristic Evaluations

Figures 1 and 2 show that one possible explanation for the poor performance of SGMPCS-H_1 is the low fitness-distance correlation. The results of the experiment that manipulated the FDC demonstrated that the performance of SGMPCS is sensitive to the FDC, at least in an artificial setting. In this section, we develop two new heuristic evaluation functions. Our primary goal is to demonstrate that in a non-artificial setting the FDC induced by the heuristic evaluation function has an impact on the search performance of SGMPCS. Our secondary goal is to improve the performance of SGMPCS.

The intuition behind both of the new heuristic evaluation functions is to include additional knowledge about the quality of the solution. In particular, we wish to create a finer heuristic evaluation that is able to better distinguish among the elite solutions (i.e., we would like fewer of the elite solutions to have a heuristic evaluation of zero

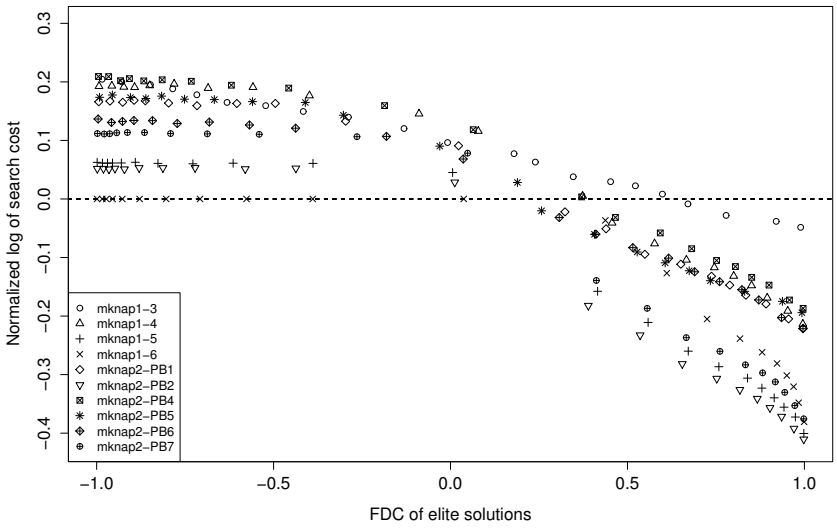

Fig. 2. A scatter-plot of the measured fitness-distance correlation versus the normalized log of search cost for the artificial heuristic evaluation in Equation (2). The low positive values of search cost for high negative FDC stem from problem instances (and settings of α and β) for which SGMPCS could not find a solution. The graph does not contain the results for problem instances mknap1-0 and mknap1-2 as they are trivially solved.

than with the H_1 function). Our main goal in proposing these heuristics is to evaluate the relationship between FDC and search performance. We expect that these heuristics will have a different FDC and wish to test if this leads to a difference in performance.[4]

$\boldsymbol{H_2}$ Recall (Section 2.3) that our CSP model of the multi-dimensional knapsack assumed that the value of the most profitable knapsack, P^*, is known. This knowledge is used in the constraint, $P = P^*$, but not otherwise exploited above. Here, we define $H_2 = |P^* - P|$.

$\boldsymbol{H_3}$ Some preliminary experiments showed that even with H_2, the elite pool often stagnated on a set of elite solutions with a zero heuristic evaluation that break one or more constraints. Therefore, in order to further refine the heuristic evaluation, we choose to use the number of broken constraints as a tie-breaker: $H_3 = H_2 + |V|$ where $|V|$ is the number of constraints violated by the (partial) assignment.

It should be noted that the only difference among the H_1, H_2, and H_3 models is the heuristic evaluation function. In particular, the constraint model is identical in all

[4] It does not seem likely that either of these heuristics will be useful, in general, for solving multi-dimensional knapsack problems because both make use of knowledge of the value of the most profitable knapsack.

three models. We now solve each of the problem instances 1000 times (with different random seeds) with each heuristic evaluation function. The other experimental details are the same as in Section 4.3.

Results Figure 3 displays the scatter plot of the normalized search performance versus FDC for all of our heuristic evaluation functions together with the minimum mean squared error line. As above, we do not include mknap1-0 and mknap1-2. Although limited by our small number of instances, the plot shows a trend of better search cost with higher FDC values. The r^2 value is 0.438 ($r = -0.662$).

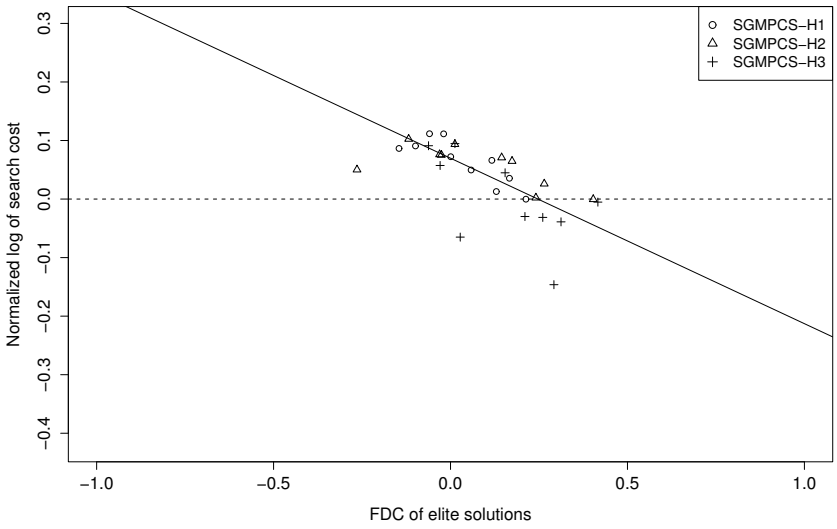

Fig. 3. A scatter-plot of the measured fitness-distance correlation versus the normalized log of search cost for three heuristic evaluation functions: H_1, H_2, H_3. The graph does not contain the results for problem instances mknap1-0 and mknap1-2 as they are trivially solved.

Table 2 displays the performance of each SGMPCS variation. For completeness we repeat the results for chronological backtracking and SGMPCS-H_1 from Table 1. While not clearly superior, SGMPCS-H_3 is competitive with *chron* overall. For the harder instances (i.e., mknap1-5, mknap2-PB2, mknap2-PB6,and mknap2-PB7 where *chron* has a high number of fails) SGMPCS-H_3 is 1.5 to 8 times better than *chron* in terms of the number of fails.

5 Discussion

In this paper, we addressed the question of developing an understanding of why SGMPCS performs as it does on constraint satisfaction problems. We demonstrated that the

	chron			SGMPCS-H_1			SGMPCS-H_2			SGMPCS-H_3		
	%sol	fails	time	%sol	fails	time	%sol	fails	time	%sol	fails	time
mknap1-4	100	15123	0.4	100	44260	1.2	100	29895	0.9	100	**11349**	0.5
mknap1-5	100	3271555	**67.2**	71	5573573	137.0	77	4839688	126.2	98	**1824457**	71.6
mknap2-PB1	100	**15223**	0.3	100	28770	0.6	100	28405	0.6	100	23445	0.7
mknap2-PB2	100	3088092	**54.1**	88	3741187	77.8	92	3191853	71.2	98	**1933160**	60.9
mknap2-PB4	100	**10167**	0.1	100	28406	0.4	100	24112	0.4	100	24370	0.5
mknap2-PB5	100	**7011**	0.1	100	15077	0.3	100	13747	0.3	100	11650	0.4
mknap2-PB6	100	16050	1.9	100	25954	3.4	100	26082	3.4	100	**8554**	1.8
mknap2-PB7	100	1472499	138.7	86	4113704	423.6	86	4287571	447.7	100	**184443**	**32.8**

Table 2. Comparison of multi-dimensional knapsack results for chronological backtracking (*chron*), and SGMPCS using the three heuristic evaluation functions H_1, H_2, H_3.

correlation between the heuristic evaluation of an elite solution and its distance to the satisfying solution, is well-correlated with the search performance of SGMPCS. Standard constructive search approaches such as chronological backtracking, randomized restart, and limited discrepancy search make no use of such heuristic information.

5.1 Limitations

There are a number of limitations to the study in this paper. First, it is a case study of 12 problem instances of one type of problem. While we believe these results are likely to be observed for other problems instances and types, a larger study is needed. Second, the poor performance of randomized restart on the multi-dimensional knapsack problems suggests that they do not exhibit heavy-tails. We believe (and have some preliminary empirical evidence on job shop scheduling problems) that SGMPCS is able to exploit heavy-tails in the same way as randomized restart. This would not be surprising since SGMPCS is a restart-based algorithm. Therefore, a full descriptive model of SGMPCS must address the impact of heavy-tailed distributions. In fact, one of the reasons that the multi-dimensional knapsack problem was chosen for this case study was precisely because we did not have to address the impact of heavy-tailed distributions. Third, it should be acknowledged that multi-dimensional knapsack problems are strange CSPs since the underlying problem is an optimization problem and we exploit this in formulating the new heuristic evaluation functions in Section 4.4. Our original motivation for choosing to apply SGMPCS to a CSP version multi-dimensional knapsack was simply because [15] did so and showed poor performance for randomized restart. Given the relationship between randomized restart and SGMPCS, this appeared to be a fertile choice. There remains some uncertainty regarding the application of the FDC-based descriptive model of SGMPCS performance on more "natural" CSPs. Nonetheless, our model makes clear, testable hypotheses that can be evaluated in future work. Finally, as a descriptive model, the work in this paper does not, on its own, produce a clear benefit for constraint solvers. That is, we have not demonstrated any improvement on the state-of-the-art for any problem classes. That was not our aim in this paper. What we have done is provided a deeper understanding of the performance of SGMPCS and a potential new source of search guidance for CP search.

5.2 Moving Toward Soft Constraints

It was noted in Section 2.1 that an alternative way to apply SGMPCS to constraint satisfaction problems is to adopt a soft constraint framework. The work in this paper makes the prediction that the success of such an approach depends, at least partially, on achieving a high correlation between the "cost" of a solution that breaks some constraints and the distance of that solution from a satisfying (or optimal in the case of MAX-CSP) solution. Such work is an important test of the generality of the results presented here.

Another approach to the incorporation of soft constraints is to define the heuristic evaluation function to be based on a soft constraint model while the primary search is done within a crisp constraint model as above. That is, when the constructive search finds a potential elite solution, the evaluation of that solution could be done using a soft constraint model. The assignments of the elite solution could be extended to find a complete assignment that minimizes the cost of the broken constraints. That cost is then used as the heuristic evaluation of the corresponding elite solution.

6 Conclusion

In this paper, the first steps were taken in understanding the search behaviour of Solution-Guided Multi-Point Constructive Search (SGMPCS). Using a constraint satisfaction model of the multi-dimensional knapsack problem, a descriptive model of SGMPCS search behaviour was developed using fitness-distance analysis, a technique common in the metaheuristic literature [10]. Empirical results, both in an artificial context and using three different heuristic evaluation functions, demonstrated that the correlation between the heuristic evaluation of a state and its proximity to the satisfying solution has a strong impact on search performance of SGMPCS. This (partial) descriptive model is important for three main reasons:

1. It makes strong, testable predictions about the behaviour of SGMPCS on other constraint satisfaction and optimization problems.
2. It provides a clear direction for improving SGMPCS search performance: the creation of, perhaps domain-dependent, heuristic evaluation functions for partial search states that are well-correlated with the distance to the nearest solution.
3. It re-introduces a heuristic search guidance concept to the constraint programming literature. Though guidance by heuristic evaluation of search states is common in metaheuristics, general AI search (e.g., A^* and game playing), and best-first search approaches, it does not appear to have been exploited in constructive, CP search. We believe this is an important direction for further investigation.

Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada, the Canadian Foundation for Innovation, the Ontario Research Fund, Microway, Inc., and ILOG, S.A..

References

1. J. C. Beck. Multi-point constructive search. In *Proceedings of the Eleventh International Conference on Principles and Practice of Constraint Programming (CP05)*, pages 737–741, 2005.
2. J. C. Beck and J.-P. Watson. Adaptive search algorithms and fitness-distance correlation. In *Proceedings of the Fifth Metaheuristics International Conference*, 2003.
3. J.C. Beck. Solution-guided multi-point constructive search for job shop scheduling. *Journal of Artificial Intelligence Research*, 2007. accepted for publication.
4. N. Beldiceanu and T. Petit. Cost evaluation of soft global constraints. In J.-C. Régin and M. Rueher, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 3011 of *LNCS*, pages 80–95. Springer-Verlag, 2004. ISBN 3-540-21836-X.
5. M. Dorigo, G. Di Caro, and L.M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.
6. S. P. Fekete, J. Schepers, and J. C. van der Veen. An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, 2007. in press.
7. I. P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. The constrainedness of search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, volume 1, pages 246–252, 1996.
8. C.P. Gomes, C. Fernández, B. Selman, and C. Bessière. Statistical regimes across constrainedness regions. *Constraints*, 10(4):317–337, 2005.
9. I. Heckman and J. C. Beck. An empirical study of multi-point constructive search for constraint satisfaction. In *Proceedings of the Third International Workshop on Local Search Techniques in Constraint Satisfaction*, 2006.
10. H.H. Hoos and T. Stüzle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2005.
11. J. Huang. The effect of restarts on the efficiency of clause learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI07)*, pages 2318–2323, 2007.
12. T. Hulubei and B. O'Sullivan. The impact of search heuristics on heavy-tailed behaviour. *Constraints*, 11(2–3):159–178, 2006.
13. R. Korf. Optimal rectangle packing: New results. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS04)*, pages 142–149, 2004.
14. M. Luby, A. Sinclair, and D. Zuckerman. Optimal speedup of Las Vegas algorithms. *Information Processing Letters*, 47:173–180, 1993.
15. P. Refalo. Impact-based search strategies for constraint programming. In *Proceedings of the Tenth International Conference on the Principles and Practice of Constraint Programming (CP2004)*, pages 557–571, 2004.
16. W. Ruml. *Adaptive tree search*. PhD thesis, Dept. of Computer Science, Harvard University, 2002.
17. B. M. Smith and M. E. Dyer. Locating the phase transition in constraint satisfaction problems. *Artificial Intelligence*, 81:155–181, 1996.
18. W.-J. van Hoeve, G. Pesant, and L.-M. Rousseau. On global warming: Flow-based soft global constraints. *Journal of Heuristics*, 12(4-5):347–373, 2006.
19. J.-P. Watson. *Empirical Modeling and Analysis of Local Search Algorithms for the Job-Shop Scheduling Problem*. PhD thesis, Dept. of Computer Science, Colorado State University, 2003.