Exploiting fitness distance correlation of set covering problems

Finger, M., Stützle, T. and **Lourenço, H.R.** (2002), Exploiting fitness distance correlation of set covering problems. *In Applications of Evolutionary Computing, S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G.R. Raidl (Eds.)* *Lecture Notes in Computer Science*, *2279:61-71*.

ISSN: 0302-9743.
Link to publication

Exploiting Fitness Distance Correlation of Set Covering Problems

Markus Finger[1], Thomas Stützle[1], and Helena Lourenço[2]

[1] Darmstadt University of Technology, Intellectics Group,
Alexanderstr. 10, 64283 Darmstadt, Germany
[2] Universitat Pompeu Fabra, Department of Economics and Business,
R. Trias Fargas 25-27, 08005 Barcelona, Spain

Abstract. The set covering problem is an $\mathcal{NP}$-hard combinatorial optimization problem that arises in applications ranging from crew scheduling in airlines to driver scheduling in public mass transport. In this paper we analyze search space characteristics of a widely used set of benchmark instances through an analysis of the fitness-distance correlation. This analysis shows that there exist several classes of set covering instances that show a largely different behavior. For instances with high fitness distance correlation, we propose new ways of generating core problems and analyze the performance of algorithms exploiting these core problems.

1 Introduction

The set covering problem (SCP) is a well-known $\mathcal{NP}$-hard combinatorial optimization problem. Its importance lies in the large number of real applications ranging from crew scheduling in airlines [1], driver scheduling in public transportation [2], and scheduling and production planning in several industries [3] that can be modeled as SCPs.

The SCP consists in finding a subset of columns of a zero-one $m \times n$ matrix such that it covers all the rows of the matrix at minimum cost. Let $M = \{1, 2, .., m\}$ and $N = \{1, 2, .., n\}$ be, respectively, the set of rows and the set of columns. Let $A = (a_{ji})$ be a zero-one matrix and $c = (c_i)$ be a n-dimensional integer vector that represents the cost of column i. We say that a column i covers a row j if $a_{ji} = 1$. The problem can be formally stated as

$$\begin{array}{ll} \text{Minimize} & v(SCP) = \sum_{i=1}^{n} c_i x_i \\ \text{s.t.} & \sum_{i=1}^{n} a_{ji} x_i \geq 1, \;\; j = 1, ..., m, \\ & x_i \in \{0, 1\}, \;\; i = 1, ..., n, \end{array}$$

where $x_i = 1$ if the column is in the solution, and $x_i = 0$, otherwise.

Because of ist enormous practical relevance, a large number of solution approaches, including exact and approximate algorithms, were proposed. Exact algorithms can solve instances with up to a few hundred rows and few thousand columns; a comparison of exact algorithms can be found in [4]. Because of the large size of SCP instances, for which exact algorithms are not anymore feasible, a large number of approximate algorithms was proposed. While pure greedy

S. Cagnoni et al. (Eds.): EvoWorkshops 2002, LNCS 2279, pp. 61–71, 2002.

construction heuristics perform rather poorly, Lagrangian-based heuristics are very popular with the most efficient ones being the approach by Ceria, Nobili and Sassano [5] and the CFT heuristic by Caprara, Fischetti and Toth [6]. Since a few years, also applications of metaheuristics to the SCP have increased. So far, the best results are mainly due to Genetic Algorithms [7,8], Simulated Annealing algorithms, the most recent being by Brusco, Jacobs and Thompson [9], and the very high performing iterated local search [10] algorithms by Marchiori and Steenbeck [11] and by Yagiura, Kishida and Ibaraki [12]. These latter two algorithms iteratively move through construction/destruction phases and local search phases.

Despite the many algorithmic approaches to the SCP, no insights into the search space characteristics of the SCP for approximate algorithms have been obtained. Yet, such insights are valuable to understand algorithm behavior as well as they may propose new ways of attacking a problem. In this article, we analyze search space characteristics of the SCP by analyzing the fitness distance correlation [13] and show that very strong differences among the search space characteristics exist between different types of instances. Based on our insights from the analysis, we propose new ways of generating core problems, that is a much smaller SCP containing a subset of the columns that are most likely to appear in optimal solutions. Experimental results with a known SA algorithm [9] that was modified to work on our core problems proves the viability of our approach leading to results competitive to other codes.

The paper is organized as follows. The next section presents the details of the fitness-distance analysis, Section 3 gives computational results for an algorithm using a new definition of core problems and we conclude in Section 4.

2 Fitness Distance Analysis

Central to the search space analysis of combinatorial optimization problems is the notion of *fitness landscape* [14,15]. Intuitively, the fitness landscape can be imagined as a mountainous region with hills, craters, and valleys. The performance of metaheuristics strongly depends on the shape of this search space and, in particular, on the ruggedness of the landscape, the distribution of the valleys, craters and the local minima in the search space, and the overall number of the local minima.

Formally, the fitness landscape is defined by (i) the set of all possible solutions $\mathcal{S}$, (ii) an objective function that assigns to every $s \in \mathcal{S}$ a fitness value $f(s)$, and (iii) a distance measure $d(s, s')$ which gives the distance between solutions s and s'. The fitness landscape determines the shape of the search space as encountered by a local search algorithm.

For the investigation of the suitability of a fitness landscape for adaptive multi-start algorithms like the best performing metaheuristics for the SCP [9], [11], [12], the analysis of the correlation between solution costs and the distance between solutions or to globally optimal solutions has proved to be a useful tool [16,13]. The fitness distance correlation (FDC) [13] measures the correlation of

the solution cost and the distance to the closest global optimum. Given a set of cost values $C = \{c_1, \ldots, c_m\}$ and the corresponding distances $D = \{d_1, \ldots, d_m\}$ to the closest global optimum the correlation coefficient is defined as:

$$r(C, D) = \frac{c_{CD}}{s_C \cdot s_D}, \text{where } c_{CD} = \frac{1}{m} \sum_{i=1}^{m} (c_i - \bar{c})(d_i - \bar{d})$$

where $\bar{c}$, $\bar{d}$ are the average cost and the average distance, s_C and s_D are the standard deviations of the costs and distances, respectively.

One difficulty for applying the FDC analysis to the SCP is that no straightforward distance measure exists, because of the SCP being a sub-set problem. Therefore, we rather use the closeness between solutions (based on the closeness, also a measure for the distance between solutions may be defined):

Definition 1. *Let $s, s' \in \mathcal{S}$ be two feasible solutions for an SCP instance, then we define the closeness $n(\cdot, \cdot)$ and the distance $d(\cdot, \cdot)$ between s and s' as*

$$n(s, s') = \text{number of same columns in } s \text{ and } s\text{'}$$
$$d(s, s') = \max(|s|, |s'|) - n(s, s')$$

A maximal distance $d(s, s') = \max(|s|, |s'|)$ means that both solutions do not have any column in common, if $d(s, s') = 0$ we have $s = s'$.

As a first step, we generated a number of best-known solutions for all the instances under concern using a variant of the SA of Brusco, Jacobs, and Thompson [9]. For instances where optimal solutions are not available, it is conjectured that the best-known solutions are actually optimal. In a second step, we generated 1000 locally optimal solutions, starting from random initial solutions, using the local search algorithm given in Figure 1. The function $\mathrm{SN}_1(s, j)$ performs the following steps: (i) it removes the column j from the current solution s, (ii) it completes s by iteratively choosing randomly a still uncovered row and adding a column with best value of $c_i/cv(i)$, where $cv(i)$ is the cover value of column i, that is, the number of still uncovered rows that is covered by column i, and (iii) it removes redundant columns.

For the fitness distance analysis we focused on two sets of benchmark instances. The first set, available from ORLIB at `http://mscmga.ms.ic.ac.uk/info.html`, is composed of randomly generated instances with varying density and size. Here, we only present results for the instances classes C–H, where classes C, D have $m = 400, n = 4000$ and a matrix density of 2% and 5%, respectively; classes E, F have $m = 500, n = 5000$ and a matrix density of 10% and 20%, respectively; and classes G, H have $m = 1000, n = 10000$ and a matrix density of 2% and 5%, respectively. Each class contains five instances. In these instances the column costs are uniformly distributed in the interval $[1, 100]$, each column covers at least one row, and each row is covered by at least two columns.

The second set with instances aa03-aa06 and aa11-aa20 stems from the paper of Balas and Carrera [17]. There are 14 instances with m varying from 105 to 272 and n varying from 3095 to 8661; the density of these instances are around 4%

```
Procedure IV^f_{N_1};
    s :=RandomSolution;
    while ( improvement ) do
        r := s;
        while ( r ≠ ∅ ∧ no improvement found so far ) do
            choose randomly a column j ∈ r and r := r \ {j};
            s' := SN_1(s, j);
            if (f(s') < f(s)) then s := s';
        end while;
    end while;
    return s;
end IV^f_{N_1};
```

Fig. 1. First-improvement local search procedure $IV^f_{\mathcal{N}_1}$.

for instances aa03-aa06 and around 2.6% for instances aa11-aa20; additionally, the instances differ in the range of the column weights, where weigths are either in a range from 91–3619 (instances aa03-aa06) or in the range of 35–2966.

2.1 Results on ORLIB Instances

Results on the FDC analysis are plotted in Figure 2 and detailed results are available in Table 1. In the plots the points are stratified according to the solution quality. The reason is that only few different values are possible for the solution quality. For example, the instance E4 has an optimal solution of 28 and therefore, a solution of 29 has a deviation of around 3.5% from the optimal solution.

The high values for the correlation coefficient (see Table 1) confirm the observation of a high, positive correlation between the solution quality and the closeness to optimal solutions in the plots in Figure 2. Only for one instance (C.3) the correlation coefficient is below 0.25. The solution quality of the local optima is relatively high and for some of the instances, the best local optima among the 1000 generated even matched the best known solutions. Particularly interesting are two observations. First, the average percentage closeness between the local optima and the total number of all different columns in the 1000 local optima depends strongly on the size of the instance as well as their densities: the larger the density the smaller is the average closeness among local optima, the less columns are in local optima and the less is the number of distinct columns in the 1000 local optima. Second, there is a direct relationship between problem size and local optima statistics: The larger the instance, the more distinct columns are encountered in the local optima (compare instance classes C and G and instance classes D and H–they have the same densities). Interestingly, for some instance classes (with same density) the average closeness of the local optima actually increases, like it is the case for C and G. This may indicate that these problem do not really become intrinsically harder with instance size.

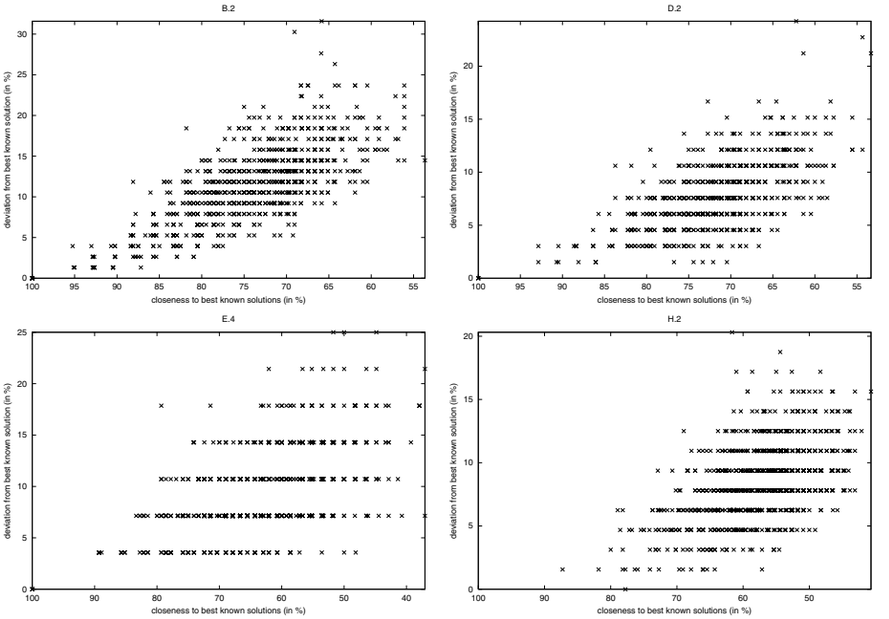

Fig. 2. Fitness distance graphs for some selected ORLIB instances.

2.2 Results on Instances from Balas and Carrera

The FDC plots of some instances from Balas and Carrera (BC) are given in Figure 3, detailed results are given in Table 2. The search space analysis of these instances shows a very different behavior. First, the closeness of the instances to the best known solutions is very low; this can be seen when inspecting the plots: all points are in the range of a closeness between 0-35%; whereas for the ORLIB instances the closeness was mostly larger than 0.5. Second, there is only a very small correlation between fitness and closeness to the best known solutions, showing that the objective function value provides much less guidance towards the global optima than on the ORLIB instances. Third, the average closeness of the local optima is very low, leading to the fact that the overall number of distinct columns in the local optima is extremely high.

These results suggest that the BC instances are much harder to solve for local search than the ORLIB instances because less guidance is given by the objective function and a much larger number of columns actually appear in good solutions, leading to a larger *effective* search space.

3 Core Problems

The results of the search space analysis can be exploited to find systematic and well justified ways for reducing instance size by defining small core problems:

generate local optima according to the algorithm of Figure 1 and define the core problem as the union of the columns contained in the local optima. Such an approach is interesting in a situation as observed for ORLIB problems, where the structure of local optima correlates strongly with that of global optima. In such a case, the columns of global optima are very likely to occur in many of the local optima.

Table 1. Results of the FDC analysis of ORLIB instances. Given are the instance identifier (PI), the best known solutions, the correlation coefficient r, the number of distinct local optima, the best, average, and worst solution found, the average closeness, the number of different columns in the 1000 local optima, and the number of best known solutions (GO) used in the FDC analysis.

PI	best known sol.	r	no. LO	sol. quality local minima			$\overline{\text{C(LO)}}$ (in %)	$\sum$ col (LO)	GO
				avg.	best	worst			
C.1	227	0.5955	762	236.67	229	272	74.58	252	1
C.2	219	0.5741	964	234.76	221	262	65.66	292	11
C.3	243	0.1355	989	266.55	252	296	64.07	310	1
C.4	219	0.6400	945	237.70	225	268	64.02	290	113
C.5	215	0.7409	870	224.48	215	270	71.02	257	7
D.1	60	0.7238	776	64.31	60	79	29.75	152	9
D.2	66	0.6306	864	70.93	66	82	31.68	166	42
D.3	72	0.4653	929	78.18	73	89	31.81	191	25
D.4	62	0.5522	664	66.73	62	79	35.61	160	3
D.5	61	0.7868	297	65.74	61	76	35.22	148	2
E.1	29	0.7202	836	30.90	29	36	17.65	119	163
E.2	30	0.2944	992	33.33	30	40	13.54	151	1
E.3	27	0.4699	962	29.92	27	35	14.39	132	1
E.4	28	0.5516	961	30.68	28	35	16.44	123	6
E.5	28	0.7602	844	30.08	28	38	17.61	126	38
F.1	14	0.6475	979	15.51	14	18	5.78	91	55
F.2	15	0.7462	868	16.26	15	19	7.22	90	98
F.3	14	0.4428	925	16.24	14	19	6.53	84	1
F.4	14	0.5802	991	15.68	14	18	5.42	90	28
F.5	13	0.2661	997	15.17	14	17	4.44	88	1
G.1	176	0.7667	995	189.31	178	204	85.93	421	66
G.2	154	0.5016	1000	165.96	158	183	82.42	404	2
G.3	166	0.4355	999	177.11	170	192	90.60	396	5
G.4	168	0.3989	1000	181.18	173	201	82.26	422	1
G.5	168	0.4897	999	181.68	173	198	81.39	439	33
H.1	63	0.3646	1000	69.57	64	77	33.11	297	1
H.2	63	0.5301	1000	69.34	64	77	32.79	298	2
H.3	60	0.4107	1000	65.90	61	77	31.57	275	1
H.4	58	0.5582	1000	63.66	59	69	31.02	277	45
H.5	55	0.6344	1000	60.25	56	66	33.58	258	45

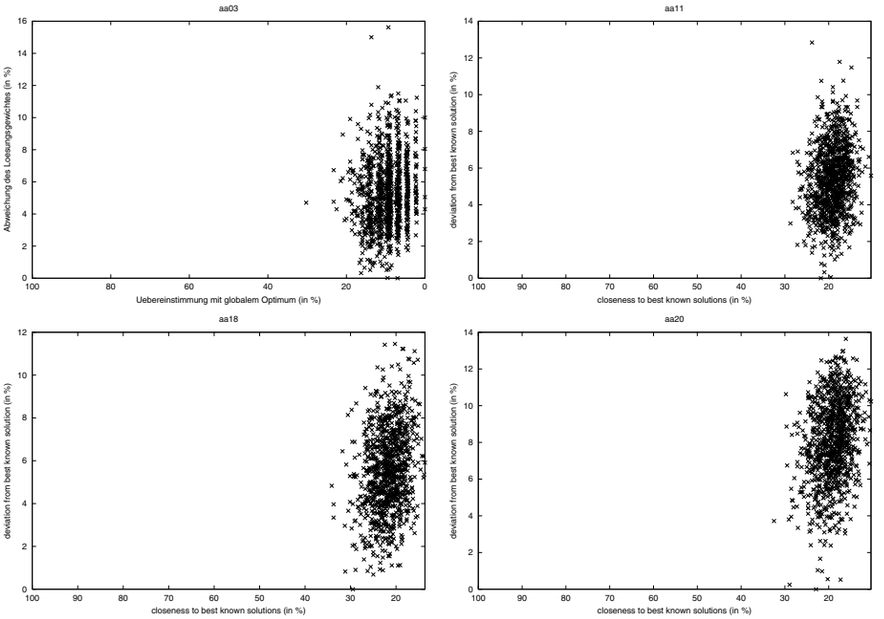

Fig. 3. Fitness distance graphs for some selected instances of Balas and Carrera.

Table 2. Results of the FDC analysis of ORLIB instances. For a description of the entries see Table 1.

PI	best known sol.	r	no. LO	sol. quality local minima			$\overline{C(LO)}$ (in %)	$\sum$ col (LO)	GO
				avg.	best	worst			
aa03	33155	0.1486	1000	36319.21	34485	39873	2.89	3565	257
aa04	34573	0.2072	1000	37650.96	34731	40803	3.89	3057	704
aa05	31623	0.0071	1000	33027.26	32235	36237	4.43	2915	155
aa06	37464	0.1262	1000	40701.95	38762	44345	4.47	2741	458
aa11	35384	0.1702	1000	39843.31	37804	42658	15.00	3733	64
aa12	30809	0.1936	1000	35385.19	32962	38114	13.82	3576	1202
aa13	33211	0.2092	1000	38143.74	36260	40836	12.04	3424	339
aa14	33219	0.2114	1000	37209.92	35441	40318	13.67	3374	48
aa15	34409	0.1943	1000	38950.44	36903	41390	14.65	3043	128
aa16	32752	0.1865	1000	37983.32	35302	40494	13.86	3137	271
aa17	31612	0.2480	1000	36608.98	34487	39334	14.26	2866	400
aa18	36782	0.2407	1000	41182.28	39013	43482	16.96	2806	1024
aa19	32317	0.2464	1000	36893.85	34395	39345	10.60	2774	4356
aa20	34912	0.2583	1000	40114.32	37116	42179	15.79	2663	55

For the generation of core problems two issues are important: First, a sufficient number of different local optima has to be generated. Second, the overall time of generating core problems and subsequently solving an instance should be smaller than solving an instance without using core problems.

We tested this idea using a variant of the Simulated Annealing algorithm by Brusco et al. [9] on the ORLIB instances which is run on the initial problem (SA) and on the core problems SA_{core}. For the generation of the core problems, we construct 25 starting solutions for the local search in Figure 1 but starting from solutions generated by a randomized greedy construction heuristic instead of random initital solutions and the core problem then contains all columns member of any of the 25 local optima. The greedy construction heuristic iteratively selects first an still uncovered row and in a second step it chooses randomly among the five highest ranked columns, where the rank of a column is determined by its cost divided by the cover value. (The use of a greedy algorithm has the advantage that the subsequent local search is much faster than from random starting solutions.)

The results of SA_{core} are in Table 4, the results of SA together with a tabu search variant ($\mathrm{IVT}^b_{\mathcal{N}_1}$) and a zero temperature SA algorithm (SA_{T_0}) are in Table 3. All results in Tables 3 and 4 are based on 25 independent trials; in the case of the results on the core problems, for each trial a new core problem

Table 3. Computational results with Tabu Search, Simulated Annealing, and zero temperature annealing.

PI	best known sol.	$IVT^b_{\mathcal{N}_1}$			SA			SA_{T_0}		
		average LG	no. opt	time	average LG	no opt	time	average LG	no opt	time
E.1	29	29	25	11.69	29	25	1.60	29	25	0.53
E.2	30	30.84	4	147.42	30	25	0.26	30	25	0.24
E.3	27	27.08	23	325.86	27	25	0.22	27	25	0.21
E.4	28	28	25	249.21	28	25	7.47	28	25	8.06
E.5	28	28	25	50.94	28	25	2.07	28	25	1.35
F.1	14	14	25	120.61	14	25	4.85	14	25	6.36
F.2	15	15	25	19.03	15	25	0.50	15	25	1.78
F.3	14	14	25	421.88	14	25	39.17	14	25	64.40
F.4	14	14	25	212.54	14	25	12.49	14	25	17.95
F.5	13	13.88	3	65.65	13.24	19	77.80	13.92	2	14.35
G.1	176	177.16	8	402.16	176	25	16.49	177.32	11	23.78
G.2	154	154.56	13	425.52	155	0	28.25	156.72	0	10.48
G.3	166	167.52	0	290.98	167.32	2	38.90	167.6	0	28.92
G.4	168	169.8	5	325.94	168.68	15	43.81	170.88	3	29.35
G.5	168	169	11	372.57	168	25	18.64	168.8	14	25.62
H.1	63	64.12	0	756.83	63.92	2	126.11	64.2	0	116.98
H.2	63	64.24	19	631.37	63.16	21	179.74	63.96	2	73.76
H.3	59	60.64	9	580.45	59.28	19	255.89	60.28	18	177.98
H.4	58	59.04	6	498.56	58	25	198.00	58.6	11	164.52
H.5	55	55.36	16	425.97	55	25	35.92	55.28	20	67.30

Table 4. Computational results with Simulated Annealing using core problems.

Problem	best known sol.	SA_{core} average sol.qual	solution quality best	no. opt.	worst	time (sec)	no. LO	$time_{LO}$	Core-problem m	No. columns
E.1	29	29	29	25	29	0.008	25	2.315	500	83.2
E.2	30	31	31	25	31	0.073	25	3.155	500	91.72
E.3	27	27.04	27	24	28	0.000	25	2.616	500	86.16
E.4	28	28	28	25	28	0.101	25	2.035	500	85.4
E.5	28	28	28	25	28	0.055	25	2.508	500	83.96
F.1	14	14	14	25	14	0.020	25	1.961	500	53.08
F.2	15	15	15	25	15	0.006	25	2.184	500	49.28
F.3	14	14	14	25	14	0.110	25	2.622	500	51.4
F.4	14	14	14	25	14	0.081	25	1.645	500	49.12
F.5	13	14	14	25	14	0.000	25	1.450	500	47.56
G.1	176	176	176	25	176	3.152	25	20.246	1000	301.48
G.2	154	155.04	154	1	156	5.776	25	17.730	1000	279.24
G.3	166	167.52	166	3	169	16.414	25	17.202	1000	283.08
G.4	168	168.48	168	19	170	11.646	25	19.640	1000	286.16
G.5	168	168.12	168	23	170	2.997	25	22.417	1000	296.32
H.1	63	64	64	25	64	2.663	25	31.347	1000	183
H.2	63	63.12	63	22	64	7.883	25	31.473	1000	179.36
H.3	59	59.56	59	15	61	6.015	25	27.839	1000	185.76
H.4	58	58.28	58	18	59	7.227	25	28.519	1000	179
H.5	55	55	55	25	55	0.537	25	28.436	1000	175.6

was generated; all algorithms were limited to 2000 iterations. The programs were coded with C++ and run on a Pentium III 700MHz CPU.[1]

The computational results show that the SA working on core problems is significantly faster, which is most visible for the largest instances of the class H. The speed-up only comes with a very minor loss in solution quality on some few instances when compared to the SA working on all columns; for some few instances even better performance could be obtained. We expect that on the few instances, where SA_{core} did not match the best-known solutions, this could be achieved by increasing the number of local optima for generating core problems. Interestingly, the largest part of the computation time is spent on the generation of the core problems and not by the SA. The main reason is certainly that the number of columns in the core problems is very small (see last column of Table 4).

[1] Let us remark here that our results (by SA or SA_{core}) to a large extent match the results obtained with the best performing algorithms for the SCP; only on some few problems slightly better average solution qualities have been reported in [9,6,11,12]. Note that the best performing approaches are using ad-hoc defined core problems that are typically modified at solution time. We refer to [6,11,12] for details.

4 Conclusions

We have analyzed some search space characteristics of the SCP and, based in this analysis we proposed new, systematic ways of generating core problems for the SCP. Computational results on the SCP instances from ORLIB showed that the resulting core problems are very small, but for the ORLIB instances they often contain all columns necessary to find the best-known solutions. In fact, some of the core problems were solved using an exact algorithm and we could verify, that the best solutions found by an Simulated Annealing algorithm working on core problems, identified consistently the optimal solutions for the core problems.

There are several ways how this work can be extended. First, we could extend our analysis to other instances from real applications. Second, faster ways of identifying core problems would be interesting, because this task consumes the largest part of the computation time of SA_{core}. Third, our approach could be enhanced by dynamically changing the core problems during the run of the algorithm as done in [6,11]. Fourth, the same ideas of generating core problems could also be applied to other problems that show significant fitness distance correlation. The results on the SCP suggest that this last idea is very promising for a number of combinatorial problems where similar high fitness distance correlations are observed.

References

1. E. Housos and T. Elmoth. Automatic optimization of subproblems in scheduling airlines crews. *Interfaces*, 27(5):68–77, 1997.
2. H. Lourenço, R. Portugal, and J.P. Paix ao. Multiobjective metaheuristics for the bus-driver scheduling problem. *Transportation Science*, 35(3):331–343, 2001.
3. F.J. Vasko and F.E. Wolf. Optimal selection of ingot sizes via set covering. *Operations Research*, 15:115–121, 1988.
4. A. Caprara, M. Fischetti, and P. Toth. Algorithms for the set covering problem. Technical Report OR-98-3, DEIS, University of Bologna, Italy, 1998.
5. S. Ceria, P. Nobili, and A. Sassano. A lagrangian-based heuristic for large-scale set covering problems. *Mathematical Programming*, 81:215–228, 1998.
6. A. Caprara, M. Fischetti, and P. Toth. A heuristic method for the set covering problem. *Operations Research*, 47:730–743, 1999.
7. J.E. Beasley and P.C. Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94:392–404, 1996.
8. A.V. Eremeev. A genetic algorithm with a non-binary representation for the set covering problem. In *Proceedings of OR'98*, pages 175–181. Springer Verlag, 1999.
9. M.J. Brusco, L.W. Jacobs, and G.M. Thompson. A morphing procedure to supplement a simulated annealing heuristic for cost- and coverage-corrleated set covering problems. *Annals of Operations Research*, 86:611–627, 1999.
10. H.R. Lourenço, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston, MA, USA, 2002. to appear.
11. E. Marchiori and A. Steenbeek. An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling. In *Real World Applications of Evolutionary Computing*, volume 1083 of *Lecture Notes in Computer Science*, pages 367–381. Springer Verlag, Berlin, Germany, 2000.

12. M. Kishida M. Yagiura and T. Ibaraki. A 3-flip neighborhood local search for the set covering problem. submitted for publication, 2001.
13. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L.J. Eshelman, editor, *Proc. of the 6th Int. Conf. on Genetic Algorithms*, pages 184–192. Morgan Kaufman, 1995.
14. P.F. Stadler. Towards a theory of landscapes. Technical Report Technical Report SFI–95–03–030, Santa Fe Institute, 1995.
15. E.D. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63:325–336, 1990.
16. K.D. Boese, A.B. Kahng, and S. Muddu. A New Adaptive Multi-Start Technique for Combinatorial Global Optimization. *Operations Research Letters*, 16:101–113, 1994.
17. E. Balas and M.C. Carrera. A dynamic subgradient-based branch and bound procedure for set covering. *Operations Research*, 44:875–890, 1996.