

Experiments on Instance Preconditioning for Combinatorial Solvers

Franc Brglez¹ and Andrej Košir²

¹ NC State University, Computer Science, Raleigh, NC 27695, USA
brglez@ncsu.edu

² University of Ljubljana, Faculty of Electrical Engineering, 1000 Ljubljana, Slovenia
andrej.kosir@ldos.fe.uni-lj.si

Abstract. Preconditioning of matrices, with the objective to solve large systems of linear equations more efficiently, is an active area of research. In contrast, there is no comparable systematic effort to precondition graph-based instances before solving them with a combinatorial solver. This paper asks the question: Does an existing preconditioning technique with known merits in solving systems of linear equations also improve the efficiency and effectiveness for a class of solvers on instances of combinatorial problems?

We propose an experimental approach to evaluate merits of the *Fiedler permutation* when solving instances of the maximal independent set (MaxIS) problem with several solvers. Preliminary results are not only encouraging, they also demonstrate the value of *Fiedler permutation* when characterizing fundamental structural properties of graph instances themselves.

Version: **2008-TR.Preconditioning_v1-Brglez**, 28-Jan-2008

1 Introduction

A recent survey paper on combinatorial scientific computing (CSC) argues that the importance of discrete algorithms in computational science will continue to grow [1]. The paper acknowledges *ordering algorithms*, a subset of combinatorial optimization problems, as an important component of sparse matrix solvers: not only are the storage requirements reduced by the orders of magnitude, so is the convergence of iterative solvers and time to find solutions with acceptable accuracy.

In contrast, there is little, if any, evidence of variable ordering strategies that have been consistently successful for solving combinatorial optimization problems more efficiently – despite the recognition of the critical importance of such variable ordering. Application domains range from the minimum set cover problem to variable ordering for minimum size Binary Decision Diagrams, one of the canonical forms for representing Boolean functions. A number of systematic and repeatable experiments with the state-of-the-art combinatorial solver *cplex* [2] demonstrate that the variable ordering remains an open issue when solving combinatorial optimization problems [3]. For example, a solver

such as *cplex* should not experience a run-time variability from 82 seconds to 1800 seconds when solving two instance of the same problem, merely rewritten in two different orders [3]. Similar large swings in performance variability of solvers other than *cplex*, all due to changing the ordering of the variables, have been reported in [4] and [5].

Preconditioning of matrices, with the objective to solve large systems of linear equations more efficiently, is an active area of research. There are numerous approaches to finding an ordering of the matrix that best satisfy a set of criteria. In contrast, there is no comparable systematic effort to precondition graph-based instances before solving them with a combinatorial solver. This paper focuses on preconditioning graph instances and evaluates the effects of such preconditioning on the efficiency and effectiveness for a class of combinatorial solvers. Specifically, we evaluate the merits of the *Fiedler permutation* when solving graph instances for the maximal independent set. Given a graph G , the permutation is based on ordering the Fiedler vector of G ; that is, the eigenvector corresponding to the second smallest eigenvalue of the Laplacian matrix of G . Properties observed with Fiedler ordering of the vertices have been found useful in graph partitioning, chromosome mapping, and matrix reordering [1, 6].

The maximal independent set can be formulated as an ILP or a graph problem. We conduct our experiments with a commercial ILP solver *cplex* [2] and three graph-based solvers that we implemented ourselves: *MaxIS_101*, *MaxIS_OpC*, and *MaxIS_MDG*. The first solver, *MaxIS_101*, uses a simple forward-pass greedy procedure that processes vertices in the natural order of vertices in the instance. The second solver *MaxIS_OpC*, uses the opportunity-cost procedure [7]; an elegant forward/backward traversal of the graph, again in the natural order of vertices in the instance. Assuming that the instance is sparse, both procedures will return the solution in time proportional to the size of the instance – however their output is sensitive to the order of vertices in the instance. We exploit this property by embedding each procedure in a simple stochastic hill-climbing solver, with repeated random pairwise swaps of rows and columns of the graph adjacency matrix, accepting only the swaps that do not decrease the value of the maximal independent set – until reaching a local optimum. The third solver is based on the minimum-degree-greedy (MDG) heuristics [8]. By its design, the expected sensitivity to the instance variable order variable is relatively low – also confirmed by our experiments.

Data sets for the experiments have been selected from various sources. We attempt to strike a balance between data sets for which we know that instances are considered random, and instances that have an underlying structure. Regardless of its origin, we consider each instance only as a reference instance for which 32 isomorphs are routinely generated as part the performance testing methodology [3, 5]. In this paper, we report results with isomorph classes based on the following reference instances:

in401.sp1: Instance *in401.sp1* is from a set of random combinatorial auction instances kindly submitted by Y. Guo [9]; with 500 vertices and 106754 edges it is not as sparse as the other instances.

frb30-15-1: Instance frb30-15-1 is from a subset of random sparse unit-weighted independent set instances with hidden solution, from <http://www.nlsde.-buaa.edu.cn/~kexu/benchmarks/setbenchmarks.htm>. It has 450 vertices and 17827 edges.

c2y: Instance c2y is from a subset of structured VLSI graph instances in the MinLA repository <http://www.lsi.upc.edu/~jpetit/MinLA/Experiments/>. It has 980 vertices and 2102 edges.

dsjc_1000.0400: Instance dsjc_1000.0400 is constructed from eight copies of a seed instance; instances are arranged in a chain with a random overlap. The random overlap is controlled such that the optimum size of the maximum independent set of the composite instance is always a simple addition of the known maximum independent set sizes of the seed instance. In our case, the seed instance dsjc125.is1 is a graph with 125 vertices 736 edges found on the Web, with comments that point to the original publications [10]. The maximum independent set size for dsjc125.is1 is 34. The composite instance has 1000 vertices and 7088 edges, and the known, but hidden maximum independent set size of dsjc_1000.0400 is $8 \cdot 34 = 272$. Some details about this construction are given in [3, 4].

The paper is organized as follows. Section 2 uses an example to illustrate various orderings: from the traditional ordering for maximal independent set to Fiedler permutations, revealing a number of graph invariants. Section 3 outlines the plan and the structure of the experiments and formulates the basic test hypothesis. Section 4 discusses the results of the experiments. Section 5 concludes the paper.

2 Background and Motivation

Independent set orderings have an important role in parallel computing, for implementing matrix solvers for both direct and iterative methods [11]. The example in Figure 1 illustrates two such orderings on two isomorphs of a 15-vertex, 21-edge graph and its adjacency matrix. Clearly, the larger the size of the independent set, the larger is the block-diagonal structure of the top-left block of the matrix.

In Figure 2 we depict four isomorphs of the earlier example; we label the top-left instance as a *reference instance* E0 and the instance to its right as an isomorph instance E0_r, related to its reference by a random permutation. In this example, a visual inspection of the adjacency matrices of each instance reveals nothing of its structure, it gives the appearance of a random matrix arrangement.

Now, consider "preconditioning" each matrix with a *Fiedler permutation*, which is different for each of the two matrices. The result is two more isomorphs, E0_f and E0_rf, shown below the respective references (E0 and E0_r). After Fiedler permutations, both adjacency matrices appear not only structured but similar.

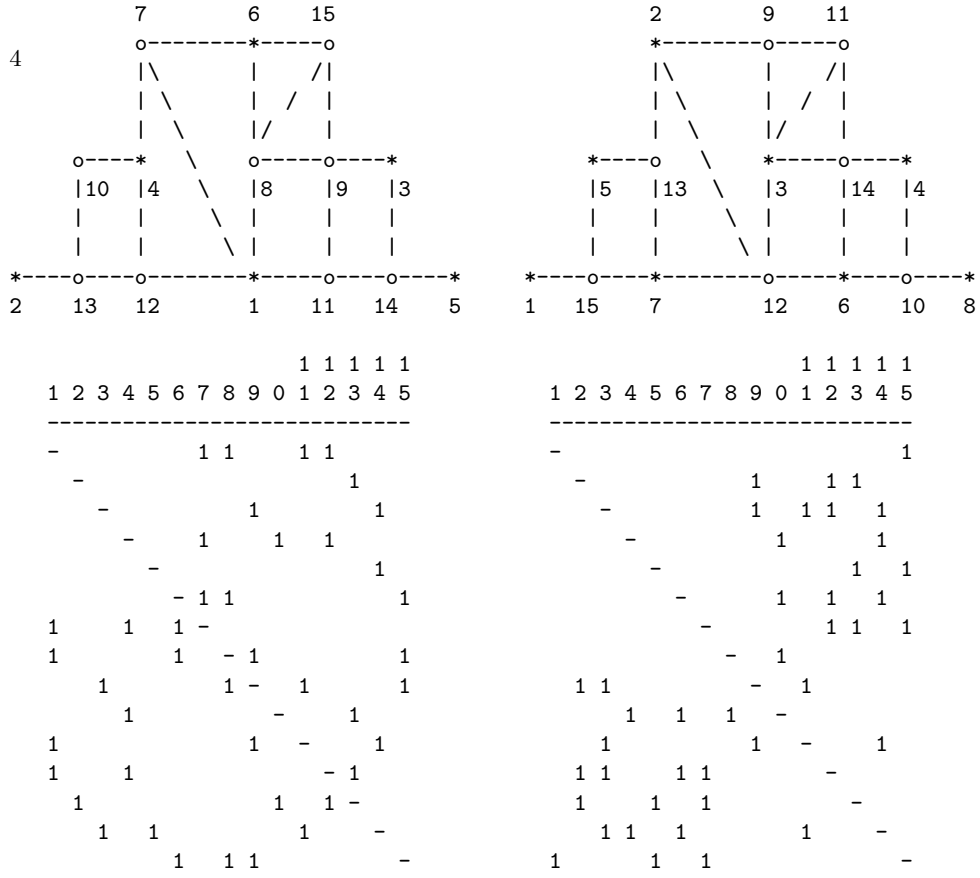


Fig. 1. Graph E0: two orderings for maximal independent sets of size 6 and 8.

There are a number of parameters that attempt to capture the quality of the matrix ordering. Some are listed in Table 1: we evaluate the bandwidth (Band-Width), envelope size (EnvSize), maximum wave-front (WaveFmax), and mean square wave-front (WaveFmsq), according to the definitions in [12]. An comprehensive survey and experiments that relate to linear arrangement (LinearArr) is available in [13, 14]. However, none of these parameters are normalized, so it is not readily apparent how 'good' the ordering really is. This motivates the introduction of a *graph correlation coefficient* (CorrCoef) that simply takes the x-y coordinates of the entries in the adjacency matrix and computes the standard-formula Pearson correlation coefficient. To our knowledge, this is the first use of the correlation coefficient that measures the quality of the order in the adjacency matrix representation of a graph. Note the that the values for CorrCoef change from 0.277 and 0.222 before preconditioning to the same vale of 0.913 after preconditioning – which is no coincidence for structured instances as we will also observe later when we consider much larger instances. Finally, the parameters IS_101 and IS_OpC represent the initial values from our solvers MaxIS_101 and MaxIS_OpC before any swapping of columns and hill-climbing takes place. The

Table 1. Observable parameter values on isomorphs of reference instance E0.

Observables	E0	E0_f	E0_r	E0_rf
BandWidth	12	6	12	6
EnvSize	69	27	75	28
WaveFmax	9	4	11	4
WaveFmsq	37.1	8.4	44.0	8.86
LinearArr	107	35	112	35
CorrCoef	0.277	0.913	0.222	0.913
IS_101	6	8	6	8
IS_OpC	6	8	6	8
MaxIS_MDG	8	8	8	8

parameter MaxIS_MDG reports the size of the independent set returned by the solver MaxIS_MDG.

In conclusion, values listed in each column of Table 1 illustrate the variability of parameters, which is significant when (1) we contrast the reference instance E0 with the isomorph E0_f based on the Fiedler permutation, and (2) we contrast the isomorph E0_r with the with the isomorph E0_rf based on another Fiedler permutation. However, while parameter variability between E0 and E0_r is to be expected, the lack of comparable variability between the isomorphs that have been 'preconditioned' with Fiedler permutations suggests that some of the listed graph parameters may be 'almost' or entirely invariant under the Fiedler permutation. This not quite what we expected, in particular for the very large graph instances in the section that follows.

3 Experimental Designs

Given the list of solvers and the list of reference instance introduced in Section 1, the generic procedure for our experiments is the executable tcl-code below:

```

proc GenericExperiment {RefFile ClassSize ClassID SolverIDs} {
  set ClassFiles [ClassGen $RefFile $ClassSize $ClassID]
  foreach SolverID $SolverIDs {
    foreach file $ClassFiles {
      set Results($ClassID,$SolverID,$file) [Encap $SolverID $file]
    }
  }
  set SummaryTables [GenTables [array get Results]]
  return $SummaryTables
}

```

There are three basic components invoked by this procedure: ClassGen, an isomorph class generator; Encap, a solver encapsulator that ensures 'standard' solver invocation and reporting of results; and GenTables, a post-processing tool that tabulates experimental results, computes statistics, and also returns latex tables such as the ones shown in the next section.

Table 2. Analysis results on instance classes of dsjc_0125, dsjc_1000_0400, and c2y.

Statistics for classes dsjc_0125 CLR (no) and dsjc_0125_CLR Fied (yes)								
Observables	Precond	RefV	MinV	MaxV	MedV	MeanV	StdV	<i>p</i> stat
WaveFmsq	no	3550	3544	3934	3721	3724	121	
	yes	3241	3241	3241	3241	3241	0.00	<i>p</i> = 0.000
LinearArr	no	29832	29832	32555	31122	31208	692	
	yes	22756	22756	22756	22756	22756	0.00	<i>p</i> = 0.000
CorrCoef	no	0.10	0.00	0.12	0.05	0.05	0.03	
	yes	0.43	0.43	0.43	0.43	0.43	0.00	<i>p</i> = 0.000
IS_101	no	26.0	22.0	29.0	26.0	25.9	1.94	
	yes	27.0	27.0	27.0	27.0	27.0	0.00	<i>p</i> = 0.000
IS_OpC	no	30.0	24.0	32.0	29.0	29.4	1.76	
	yes	30.0	30.0	30.0	30.0	30.0	0.00	<i>p</i> = 0.000
MaxIS_MDG	no	33.0	32.0	34.0	33.0	32.9	0.39	
	yes	34.0	34.0	34.0	34.0	34.0	0.00	<i>p</i> = 0.000

Statistics for classes dsjc_1000_0400_CLR (no) and dsjc_1000_0400_CLR.Fied (yes)								
Observables	Precond	RefV	MinV	MaxV	MedV	MeanV	StdV	<i>p</i> stat
WaveFmsq	no	54399	54399	250554	244103	238068	33634	
	yes	63084	62579	63084	62579	62816	256	<i>p</i> = 0.000
LinearArr	no	590740	590740	2384027	2368557	2201457	526522	
	yes	1011129	1011129	1011129	1011129	1011129	0.00	<i>p</i> = 0.000
CorrCoef	no	0.87	0.05	0.87	0.07	0.09	0.14	
	yes	0.80	0.80	0.80	0.80	0.80	0.00	<i>p</i> = 0.000
IS_101	no	208	187	212	200	199	6.33	
	yes	219	218	219	218	218	0.51	<i>p</i> = 0.000
IS_OpC	no	239	213	234	224	224	5.18	
	yes	232	232	232	232	232	0.00	<i>p</i> = 0.000
MaxIS_MDG	no	258	252	262	258	258	2.56	
	yes	262	259	262	259	260	1.52	<i>p</i> = 0.000

Statistics for classes c2y_CLR (no) and c2y_CLR.Fied (yes)								
Observables	Precond	RefV	MinV	MaxV	MedV	MeanV	StdV	<i>p</i> stat
WaveFmsq	no	60412	60412	159703	120487	124965	18180	
	yes	9002	2543	9002	2543	5167	3223	<i>p</i> = 0.000
LinearArr	no	517737	517737	725802	678390	669739	55140	
	yes	97214	97214	97214	97214	97214	0.00	<i>p</i> = 0.000
CorrCoef	no	0.36	0.15	0.36	0.20	0.20	0.04	
	yes	0.96	0.96	0.96	0.96	0.96	0.00	<i>p</i> = 0.000
IS_101	no	399	358	383	369	370	6.41	
	yes	424	416	424	424	420	4.06	<i>p</i> = 0.000
IS_OpC	no	416	394	421	403	404	5.51	
	yes	434	434	434	434	434	0.00	<i>p</i> = 0.000
MaxIS_MDG	no	450	438	450	445	444	3.16	
	yes	445	443	445	445	444	1.01	<i>p</i> = 1.000

Table 3. Analysis results on instance classes of frb30-15-1 and in401.sp1.

Statistics for classes frb30-15-1_CLR (no) and frb30-15-1_CLR_Fied (yes)								
Observables	Precond	RefV	MinV	MaxV	MedV	MeanV	StdV	p stat
WaveFmsq	no	56017	56017	65090	64539	64336	1535	
	yes	53476	53476	53476	53476	53476	0.00	$p = 0.000$
LinearArr	no	2221475	2221475	2695072	2673278	2632240	134874	
	yes	1909043	1909043	1909043	1909043	1909043	0.00	$p = 0.000$
CorrCoef	no	0.21	-0.01	0.21	0.01	0.02	0.04	
	yes	0.42	0.42	0.42	0.42	0.42	0.00	$p = 0.000$
IS_101	no	22.0	18.0	23.0	20.0	20.0	1.03	
	yes	24.0	24.0	24.0	24.0	24.0	0.00	$p = 0.000$
IS_OpC	no	22.0	20.0	24.0	22.0	22.0	1.18	
	yes	25.0	25.0	25.0	25.0	25.0	0.00	$p = 0.000$
MaxIS_MDG	no	22.0	23.0	27.0	24.0	24.4	0.91	
	yes	26.0	26.0	26.0	26.0	26.0	0.00	$p = 0.000$

Statistics for classes in401.sp1_CLR (no) and in401.sp1_CLR_Fied (yes)								
Observables	Precond	RefV	MinV	MaxV	MedV	MeanV	StdV	p stat
WaveFmsq	no	83498	83089	83540	83462	83400	154	
	yes	81943	81943	82532	81943	82201	297	$p = 0.000$
LinearArr	no	16985538	16985538	16985538	16985538	16985538	0.00	
	yes	1011129	1011129	1011129	1011129	1011129	0.00	$p = 0.000$
CorrCoef	no	0.00	0.00	0.00	0.00	0.00	0.00	
	yes	0.00	0.00	0.00	0.00	0.00	0.00	$p = 0.000$
IS_101	no	8.00	5.00	11.0	8.00	7.88	1.39	
	yes	12.0	9.00	12.0	12.0	10.6	1.52	$p = 0.000$
IS_OpC	no	12.0	9.00	13.0	11.0	11.1	0.96	
	yes	12.0	12.0	12.0	12.0	12.0	0.00	$p = 0.000$
MaxIS_MDG	no	13.0	13.0	14.0	13.0	13.3	0.48	
	yes	14.0	13.0	14.0	14.0	13.5	0.51	$p = 0.085$

For each reference instance we generate an instance class *_CLR of 32 isomorphs (CLR is a legacy acronym for random column, local, row permutation). For each instance in class *_CLR, we compute a Fiedler permutation which we use to generate an instance in class *_CLR_Fied.

Experiments are performed by each solver on instances from both *_CLR and *_CLR_Fied class and the main objective of experiments in this design is to test the following hypothesis:

For the same reference instance and the same solver,
the isomorph class CLR is equivalent to the isomorph class CLR_Fied.

Our expectation is that we will show, with statistical significance, that we shall *reject* this hypothesis, i.e. for most test cases we shall demonstrate that (1) the Fiedler-based preconditioning matters and (2), when it does matter, it does not necessarily imply an 'improvement'. *In this paper, we make no explicit attempt to rank the performance of a solver vis-a-vis another solver.*

Our inferences are based on observations of mean values of subset of variables introduced in Table 1: WaveFmsq, LinearArr, CorrCoeff, IS_101, IS_OpC, and MaxIS_MDG. In addition, when we invoke the stochastic optimizers MaxIS_101 and MaxIS_OpC we observe two variables: independent set size returned by optimizers (ObjectiveBest) and the total number of stochastic column-row swaps in the graph adjacency matrix (Trials).

When we run cplex, we observe the value of ObjectiveBest only; its run terminates, if needed, by an external interrupt at 64 seconds for each instance. In contrast, the two stochastic solvers MaxIS_101, MaxIS_OpC terminate on their own in less than 64 seconds per instance. The computing platform is an Intel-based processor, 3.2 GHz, 2 GB cache, under RedHat Linux.

4 Discussion of Results

We present results in two groups of tables. Tables 2 and 3 generalize the motivational experiment in Table 1 on instances of the isomorph class E0. We now report statistics for variables WaveFmsq, LinearArr, CorrCoeff, IS_101, IS_OpC, and MaxIS_MDG that were observed on a total of 10 isomorph classes, each with 33 instances (including the reference). Five of these classes refer to isomorphs generated under random permutation from its reference instance (rows labeled as 'no' in each table). The additional five classes refer to isomorphs from the first group, re-written under the Fiedler permutation (rows labeled as 'yes' in each table). Classes based on dsjc_1000_0400 and c2y represent isomorphs of structured instances; classes based on dsjc_0125, frb30-15-1 and in401_sp1 represent isomorphs of random reference instances.

Tables 4 and 5 extend the experiments on the same 8 isomorph classes described above (class of dsjc_0125 has been dropped since it represent an optimization problem that is 'too easy'). The variables we now observe are Trials and ObjectiveBest, as reported by the stochastic solvers MaxIS_101 and MaxIS_OpC. These solvers terminate when reaching a local optimum, reporting the total number of trials and the best value of the maximum independent set. As explained in the earlier section, the branch-and-bound ILP solver cplex will terminate by external interrupt after 64 seconds, reporting the best value of the maximum independent set, without proving whether it represents the optimum. For the class of c2y it does prove (easily) that the value of 454 is indeed the maximum value of the independent set.

All results Tables 2–5 are reported under the same schema. Columns of statistics can be readily interpreted. The last column, on p -value statistics, relates to data in the adjacent 'no/yes' rows and requires a special attention. Each value of p is computed from statistics using a t -test. If we observe the value of $p > 0.05$, we should accept the hypothesis that the respective means on the two adjacent rows are equivalent, i.e. accept design hypothesis stated in the preceding section. Whenever we reject this hypothesis, we can resolve whether difference induced by the Fiedler permutation has improved or degraded the value of the parameter under observation. In the space available, we now highlight some of our observations in two groups, beginning with results in Tables 2 and 3:

Table 4. Optimization results on instance classes of dsjc_1000_0400 and c2y.

Statistics for classes dsjc_1000_0400_CLR (no) and dsjc_1000_0400_CLR_Fied (yes)

Solver	Precond	Observed	RefV	MinV	MaxV	MedV	MeanV	StdV	p stat
MaxIS_101	no	Trials	41492	22315	63132	38161	38467	11092	
	yes	Trials	31299	26392	31299	26392	28692	2488	$p = 0.000$
	no	ObjBest	269	266	272	270	270	1.50	
	yes	ObjBest	269	267	269	267	268	1.01	$p = 0.000$
MaxIS_OpC	no	Trials	33646	16101	33646	31199	27995.7	6188.1	
	yes	Trials	45738	45738	45738	45738	45738	0	$p = 0.000$
	no	ObjBest	272	268	272	271	270.7	1.42	
	yes	ObjBest	270	270	270	270	270	0	$p = 0.000$
cplex090	no	ObjBest	248	232	251	244	243	5.10	
	yes	ObjBest	239	234	247	241	242	3.81	$p = 0.328$

Statistics for classes c2y_CLR (no) and c2y_CLR_Fied (yes)

Solver	Precond	Observed	RefV	MinV	MaxV	MedV	MeanV	StdV	p stat
MaxIS_101	no	Trials	20290	14732	33315	19722	21337	5607	
	yes	Trials	18417	17310	30559	24278	24183	5530	$p = 0.000$
	no	ObjBest	449	448	454	452	451	1.57	
	yes	ObjBest	452	452	453	452	452	0.46	$p = 0.001$
MaxIS_OpC	no	Trials	14743	13971	42161	18780	20793	7524	
	yes	Trials	19042	14803	19042	17680	17396	1370	$p = 0.020$
	no	ObjBest	453	449	454	452	452	1.23	
	yes	ObjBest	454	452	454	453	453	0.44	$p = 0.060$
cplex090	no	ObjBest	454	454	454	454	454	0.00	
	yes	ObjBest	454	454	454	454	454	0.00	$p = 1.000$

dsjc_*: The reference instance dsjc_0125 is the 'building block' of dsjc_1000_0400 which contains 8 overlapping instances of dsjc_0125; its construction is outlined in Section 1. Instance dsjc_0125 is known as a random instance with CorrCoef at 0.43 after Fiedler permutation (with the mean of 0.05 before the Fiedler permutation). However, a well-defined structure is revealed for dsjc_1000_0400: the correlation coefficient mean increases from 0.09 to 0.80 after the Fiedler permutation, and *remains invariant at 0.80 under the Fiedler permutation*. Still, the value of 0.80 is not the highest, the highest value of CorrCoef (0.87) is achieved with the reference instance *before* the Fiedler permutation, reflecting the 'good order' in which the reference instance was constructed!

$p > 0.05$: The hypothesis that Fiedler-based preconditioning makes no difference must be accepted when we apply MaxIS_MDG solver to instance classes of c2y and in401_sp1. In all other 28 cases examined in these tables, we must reject this hypothesis; furthermore, in all 28 cases, the effects of Fiedler-based preconditioning are beneficial.

invariance: Fiedler-based preconditioning reduces the variance of observed parameters. We summarize it as a ratio of entries with non-zero variance versus the total of 5: 3/5 for WaveFmsq, 0/5 for LinearArr, 0/5 for CorrCoef, 3/5 for IS_101, 0/5 for IS_OpC, and 3/5 for MaxIS_MDG.

Table 5. Optimization results on instance classes of frb30-15-1 and in401_sp1.

Statistics for classes frb30-15-1_CLR (no) and frb30-15-1_CLR.Fied (yes)									
Solver	Precond	Observed	RefV	MinV	MaxV	MedV	MeanV	StdV	p stat
MaxIS_101	no	Trials	6596	4493	12888	5728	6301	2003	
	yes	Trials	5971	5971	5971	5971	5971	0.00	$p = 0.000$
	no	ObjBest	27.0	24.0	28.0	25.0	25.5	0.95	
	yes	ObjBest	26.0	26.0	26.0	26.0	26.0	0.00	$p = 0.000$
MaxIS_OpC	no	Trials	4785	4027	8891	5693	6172	1801	
	yes	Trials	4175	4175	4175	4175	4175	0.00	$p = 0.000$
	no	ObjBest	27.0	25.0	28.0	26.5	26.4	0.80	
	yes	ObjBest	27.0	27.0	27.0	27.0	27.0	0.00	$p = 0.000$
cplex090	no	ObjBest	22.0	20.0	27.0	24.5	24.3	1.51	
	yes	ObjBest	25.0	24.0	26.0	24.0	24.3	0.58	$p = 0.752$

Statistics for classes in401_sp1_CLR (no) and in401_sp1_CLR.Fied (yes)									
Solver	Precond	Observed	RefV	MinV	MaxV	MedV	MeanV	StdV	p stat
MaxIS_101	no	Trials	10514	5205	11456	7465	7765	1710	
	yes	Trials	4612	4612	8718	4612	6408	2069	$p = 0.002$
	no	ObjBest	14.0	11.0	14.0	13.0	13.1	0.91	
	yes	ObjBest	13.0	13.0	14.0	13.0	13.5	0.51	$p = 0.051$
MaxIS_OpC	no	Trials	5425	4482	12444	5597	6119.6	1815.6	
	yes	Trials	4607	4607	4607	4607	4607	0	$p = 0.000$
	no	ObjBest	13	13	15	14	13.8	0.56	
	yes	ObjBest	14	14	14	14	14	0	$p = 0.000$
cplex090	no	ObjBest	9.00	9.00	13.0	11.0	11.4	1.10	
	yes	ObjBest	12.0	9.00	12.0	12.0	10.6	1.52	$p = 0.049$

Highlights for results in Tables 4 and 5 include:

$p > 0.05$: The hypothesis that Fiedler-based preconditioning makes no difference must be accepted when we apply cplex solver in 3 out of 4 instance classes in these two tables. There is 1 case out of 4 for both MaxIS_101 and MaxIS_OpC where we must accept the equal means hypothesis for ObjectiveBest. However, Fiedler-based preconditioning does make a difference when we observe the value of Trials reported by both MaxIS_101 and MaxIS_OpC – but the value increases only in 1 from a total of 8 cases.

***invariance*:** Fiedler-based preconditioning reduces the variance of both Trials and ObjectiveBest reported by MaxIS_101 and MaxIS_OpC. We summarize it as a ratio of entries with non-zero variance wrt to the total of 8: 4/8 for Trials and 4/8 for ObjectiveBest.

5 Conclusions

The initial investigation of variable orderings was motivated by the objective to improve the stochastic performance of the solver MaxIS_OpC. Initial results demonstrate that Fiedler permutation of the adjacency matrix does improve the performance of MaxIS_OpC while also reducing its variability. A comparison with a faster preconditioner such as 'metis' [15, 16] may be the next step. Perhaps the more unexpected by-product of this research are the findings that the

Fiedler permutation induces invariance of several graph metrics such as linear arrangements as well as the correlation coefficient, also introduced in this paper. The latter, in combination with Fiedler permutation, may well be the most important discovery in this paper.

References

1. B. Hendrickson and A. Pothen. Combinatorial scientific computing: The enabling power of discrete algorithms in computational science. *Lecture Notes in Computer Science 4395*, pages 260–280, 2007.
2. Home page for cplex, 2007. See <http://www.ilog.com/products/cplex/>.
3. F. Brglez and J. A. Osborne. Performance testing of combinatorial solvers with isomorph class instances. In *ACM-FCRC, 2007*. Proc. of Workshop on Experimental Computer Science, San Diego, 13-14 June 2007. Published under <http://portal.acm.org/>.
4. M. F. M. Stallmann and F. Brglez. High-contrast algorithm behavior: Observation, conjecture, and experimental design. In *ACM-FCRC, 2007*. Proc. of Workshop on Experimental Computer Science, San Diego, 13-14 June 2007. Published under <http://portal.acm.org/>.
5. F. Brglez, X. Y. Li, and M. F. M. Stallmann. On SAT instance classes and a method for reliable performance experiments with SAT solvers. *Ann. Math. Artif. Intell.*, 43(1):1–34, 2005.
6. B. Mohar. The Laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, pages 871–898, 1991.
7. K. Akcoglu, J. Aspnes, B. DasGupta, and M.-Y. Kao. Opportunity cost algorithms for combinatorial auctions. In E. J. Kontogiorghes and B. Rustem and S. Siokos, editor, *Applied Optimization: Computational Methods in Decision-Making, Economics and Finance*. Kluwer Academic Publishers, 2002.
8. M. M. Halldorsson and J. Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. In *STOC, 1994*. In Proc. of 26th Symposium on Theory of Computing.
9. Y. Guo, A. Lim, B. Rodrigues, and Y. Zhu. Heuristics for a bidding problem. *Comput. Oper. Res.*, 33(8):2179–2188, 2006.
10. D. S. Johnson, R. Aragon C, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partitioning. *Operations Research*, 39:378–406, 1991.
11. Y. Saad. *Iterative Methods for Sparse Linear Systems, Second Edition*. SIAM, 2003.
12. G. Kumfert and A. Pothen. Two improved algorithms for envelope and wavefront reduction. *BIT*, 1997.
13. J. Diaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Comput. Surv.*, 34(3):313–356, 2002.
14. J. Petit. Experiments on the minimum linear arrangement problem. *ACM Journal of Experimental Algorithms*, 8, 2003.
15. F. Karypis and V. Kumar. Metis: Unstructured graph partitioning and sparse matrix ordering, 1995. See <http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/-manual.pdf>.
16. F. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. In *Journal of International Conference on Parallel Processing*, pages 113–122, 1995.