Discrete Optimization 6 (2009) 214–219

Contents lists available at ScienceDirect

Discrete Optimization

journal homepage: www.elsevier.com/locate/disopt

Note

Optimisation of unweighted/weighted maximum independent sets and minimum vertex covers

Wayne Pullan

School of Information and Communication Technology, Griffith University, Gold Coast Campus, Gold Coast, QLD, Australia

ARTICLE INFO

Article history:
Received 16 October 2006
Received in revised form 4 December 2008
Accepted 4 December 2008
Available online 16 January 2009

Keywords:
Local search
Maximum weighted independent set
Minimum weighted vertex cover

ABSTRACT

This paper extends the recently introduced Phased Local Search (PLS) maximum clique algorithm to unweighted/weighted maximum independent set and minimum vertex cover problems. PLS is a stochastic reactive dynamic local search algorithm that interleaves sub-algorithms which alternate between sequences of iterative improvement, during which suitable vertices are added to the current sub-graph, and plateau search, during which vertices of the current sub-graph are swapped with vertices not contained in the current sub-graph. These sub-algorithms differ in their vertex selection techniques and also in the perturbation mechanism used to overcome search stagnation. PLS has no problem instance dependent parameters and achieves state-of-the-art performance over a large range of the commonly used DIMACS and other benchmark instances.

1. Introduction

Given an undirected graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ is the set of all vertices of G and $E \subseteq V \times V$ the set of all edges, an independent set S of G is a subset of V whose elements are pairwise non-adjacent. The Maximum Independent Set (MIS) problem consists of identifying the independent set S of G which has maximum cardinality. The size of the maximum independent set of G is the stability number of G and is denoted by α. A vertex cover for G is a subset T of G such that, for each edge $(u, v) \in E$, at least one of u or v belongs to T. The Minimum Vertex Cover (MVC) problem consists of identifying the vertex cover T of G which has minimum cardinality (denoted by β). The MIS and MVC problems are related in that the maximum independent set S of G contains all those vertices that are not in the minimum vertex cover T of G (i.e. $S = V - T$ and $\alpha + \beta = n$) [1]. If a positive weight w_i is associated with each vertex $i \in V$ then, for a subset $S \subseteq V$, the weight of S is defined as $W(S) = \sum_{i \in S} w_i$. The Maximum Weight Independent Set (MWIS) of G is the independent set S of G, of cardinality α_s, which maximises $W(S)$. The total weight of the MWIS of G is denoted by W_s in this study. The Minimum Weight Vertex Cover (MWVC) of G is the vertex cover T of G, of cardinality β_c, which minimises $W(T)$ and the total weight of this vertex cover is denoted by W_c. The relationships $S = V - T$ and $\alpha_s + \beta_c = n$ identified above for MIS and MVC problems also hold for MWIS and MWVC instances and, in addition, if W_t is the total weight of all the vertices in G, $W_s + W_c = W_t$.

MIS (MVC) is $\mathcal{NP}$-hard and the associated decision problem is $\mathcal{NP}$-complete [2]; therefore, large and hard instances of MIS (MVC) are typically solved using heuristic approaches, in particular, greedy construction algorithms and stochastic local search algorithms such as simulated annealing, genetic algorithms and tabu search. Although some algorithms have been empirically evaluated on benchmark instances from the Second DIMACS Challenge [3], it is difficult to compare experimental results between studies due to differences in the respective experimental protocols, benchmarks and run-time environments. In addition, as there are no widely accepted MWIS/MWVC test benchmarks, a major goal of this paper is to establish benchmark results which will allow future investigators a means of easily comparing their results with those obtained in this study.

E-mail address: w.pullan@griffith.edu.au.

1572-5286/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.disopt.2008.12.001

Existing heuristic algorithms for the MIS problem include: Continuous Based Heuristic (CBH) [4]; Optimised Crossover Heuristic (OCH) [5]; QSH [6] which uses a sophisticated greedy heuristic that first finds a local solution using a straightforward greedy approach, and then attempts to find a better solution using information provided by the stationary points obtained by optimising a quadratic function over a sphere; and the evolutionary algorithm Widest Acyclic Orientation (WAO) [7]. This study adapts the recently introduced PLS algorithm [8] to the MIS/MWIS and MVC/MWVC problems. PLS is a stochastic reactive dynamic local search algorithm that interleaves sub-algorithms which alternate between sequences of iterative improvement, during which suitable vertices are added to the current sub-graph, and plateau search, during which vertices of the current sub-graph are swapped with vertices not contained in the current sub-graph.

The remainder of this article is structured as follows: The PLS algorithm and key aspects of its efficient implementation are first described. Next, empirical performance results for the MIS/MVC benchmarks are presented and compared with MIS results from previous studies. The results for the MWIS/MWVC benchmark instances are then described and, finally, the main contributions of this work are summarised, and some directions for future research outlined.

2. The PLS algorithm

PLS interleaves three sub-algorithms which use random selection (*Random*), random selection within vertex degree (*Degree*), and random selection within vertex penalties (*Penalty*), and is now described using the following notation: $G(V, E)$ — an undirected graph with $V = \{1, 2, \ldots, n\}$, $E \subseteq \{\{i, j\} : i, j \in V\}$; $N(i) = \{j \in V : \{i, j\} \in E\}$ — the vertices adjacent to i; K — the current independent set/vertex cover of G; $W_v(K)$ – the total vertex weight of the current independent set/vertex cover; and $C_p(K) = \{i \in V : |K \setminus N(i)| = p\}$, $p = 0, 1$ — the set of all vertices not adjacent/adjacent to exactly p vertices in K.

The MIS/MVC variant of the PLS algorithm is as follows:

Algorithm PLS (*G*, *ts*, *max-selections*)
    Input: graph G; integers *ts* (target size), *max-selections*
    Output: K of size *ts* **or** 'failed'
1. $selections := 0, pu := 0, pd := 2$;
2. $sa := Random, iterations := 50$;
3. $\langle$Randomly select a vertex $v \in V, K := \{v\}\rangle$;
4. $\forall\, i \in V,\ p_i := 0$;
5. **do**
6.   **do**
7.   **while** $C_0(K) \setminus U \neq \emptyset$ **do**
8. $v := Select(C_0(K), sa)$;
9. $K := K \cup \{v\}$;
10. $selections := selections + 1$;
11.   **if** $|K| = ts \vee W_v(K) = ts$ **then return** K;
12. $U := \emptyset$;
13.   **end while**
14.   **if** $C_1(K) \setminus U \neq \emptyset$ **then**
15. $v := Select(C_1(K) \setminus U, sa)$;
16. $K := [K \cup \{v\}] \setminus \{i\}, U := U \cup \{i\}$, where $\{i\} = K \setminus N(v)$;
17. $selections := selections + 1$;
18.   **end if**;
19.   **while** $C_0(K) \neq \emptyset$ **or** $C_1(K) \setminus U \neq \emptyset$;
20. $iterations := iterations - 1$;
21. $UpdatePenalties(sa)$;
22. $Perturb(sa)$;
23. **until** $selections \geq$ *max-selections*
24. **return** 'failed';

The PLS algorithm operates as follows: after selecting an initial vertex from the given graph G uniformly at random and setting the current independent set/vertex cover K to the set consisting of this single vertex, all vertex penalties are initialised to zero. Then, the search, starting at the **do** (lines 5–23 of the algorithm; a single complete execution of lines 5–23 is referred to as an "iteration"), alternates between an iterative improvement phase, during which vertices from $C_0(K)$ are added to the current independent set/vertex cover K, and a plateau search phase, in which vertices from $C_1(K)$ are swapped with the vertex in K with which they share/do not share an edge. The search phase terminates when $C_0(K) = \emptyset$ and either $C_1(K) = \emptyset$ or all vertices that are in $C_1(K)$ have already been an element of K during the current iteration. As the final step of the iteration, a perturbation of K is performed to generate a new starting point for the search. Iterations are repeated until either the MIS/MVC is found or the number of allowed *selections* (additions to the current independent set) is exceeded. Initially, PLS performs 50 iterations of the *Random* sub-algorithm (referred to as a sub-algorithm "stage"), followed by a stage of 50 iterations of the *Penalty* sub-algorithm and then a stage of 100 iterations of the *Degree* sub-algorithm.

Within PLS, the vertex selection methods for each sub-algorithm are implemented within the function *Select* while the perturbations for each sub-algorithm and the switch to the next sub-algorithm at the completion of each stage are implemented within the function *Perturb*. Note that the differences between the sub-algorithms are wholly contained within the *Select* and *Perturb* functions. Finally, penalty updates are performed (*UpdatePenalties*) during all sub-algorithms but penalties are only used for vertex selection when the *Penalty* sub-algorithm is active. Transitioning between sub-algorithms is implemented so that the *Random* and *Degree* sub-algorithms always resume from the point at which their previous stage completed. However, the *Penalty* sub-algorithm continues from the point at which the preceding *Random* sub-algorithm stage terminated.

The sub-algorithms differ firstly in their vertex selection techniques in that selection can be solely based on randomly selecting a vertex (*Random*), randomly selecting within highest/lowest vertex degree (*Greedy*) or randomly selecting within vertex penalties that are dynamically adjusted during the search (*Penalty*). Secondly, the perturbation mechanism used to overcome search stagnation differs between the sub-algorithms. For the *Random* and *Greedy* sub-algorithms, at the completion of the iteration, function *Perturb* is invoked to uniform randomly select a vertex v, add this to K and remove all vertices from K that are connected/not connected to v. This perturbation mechanism provides for some continuity in the search and also maintains K as relatively large at all times. However, for the *Penalty* sub-algorithm, at the completion of the iteration, K is initialised to contain only a uniform randomly selected vertex v. This perturbation mechanism provides for relatively large discontinuities in the search trajectory.

3. Empirical performance results

In order to evaluate the performance and behaviour of PLS for MIS problems, extensive computational experiments were performed on the benchmark instances identified below. All experiments for this study were performed on a computer that, when executing the DIMACS Machine Benchmark[1] required 0.31 CPU seconds for r300.5, 1.93 CPU seconds for r400.5 and 7.35 CPU seconds for r500.5. Note that, in this section, only abbreviated results that summarise the performance of PLS are presented. Complete results are available at http://www.intelligent-optimization.org/OM/mis_mvc_tables.pdf.

3.1. BHOSLIB benchmark

Benchmarks with Hidden Optimum Solutions for Graph Problems (Maximum Clique, Maximum Independent Set, Minimum Vertex Cover and Vertex Coloring) (BHOSLIB).[2] These MIS/MVC benchmark instances are directly transformed from forced satisfiable SAT benchmarks, with the set of vertices and the set of edges respectively corresponding to the set of variables and the set of binary clauses in SAT instances.

To evaluate the MIS/MVC performance of PLS on the BHOSLIB benchmark instances, 100 independent trials were performed for each instance using target MIS/MVC sizes corresponding to the respective known optimal sizes. The results from these experiments are displayed in Table 1. Note that PLS finds optimal solutions with a success rate of 100% over all 100 trials per instance for 22 of the 40 BHOSLIB instances and only fails completely on one MIS instance. As would be expected, there is close correlation (correlation coefficient $= 0.9958$) between the success rates for corresponding MIS/MVC instances.

Finding optimum solutions to the BHOSLIB MIS/MVC problem instances is equivalent to finding solutions to the corresponding forced satisfiable CSP and SAT instances. Some corresponding SAT instances of BHOSLIB MIS instances were used for SAT Competition 2004 (55 SAT solvers) with the results shown in Table 1. As can be seen, the results for PLS are at least competitive with if not an improvement on these results.

3.2. DIMACS and DIMACS-C benchmark

The DIMACS benchmark consists of all 80 MC instances from the Second DIMACS Implementation Challenge (1992–1993)[3], which have also been used extensively for benchmarking purposes in the recent literature on MC algorithms. These problem instances range in size from less than 50 vertices and 1000 edges to greater than 3300 vertices and 5 000 000 edges. The DIMACS-C benchmark consists of all the complement graphs of the DIMACS benchmark (the complement graph of $G = (V, E)$ is the graph $\overline{G} = (V, \overline{E})$ where $\overline{E} = \{(i, j) \mid i, j \in V, i \neq j \text{ and } (i, j) \notin E\}$).

The putative maximum independent sets for the DIMACS benchmark were determined from a combination of long running PLS trials and shorter, more numerous trials (which also determined the putative minimum vertex covers for each instance). For the DIMACS-C benchmark, as the maximum independent set for graph $\overline{G}$ is identical to the maximum clique (MC) of G [1], the currently accepted maximum clique sizes for the 80 DIMACS MC instances [8] have been used as the

1 *dmclique*, ftp://dimacs.rutgers.edu in directory http://pub/dsj/clique.

2 http://www.nlsde.buaa.edu.cn/kexu/benchmarks/graph-benchmarks.htm.

3 http://dimacs.rutgers.edu/Challenges/.

Table 1
PLS MIS and MVC performance results, averaged over 100 independent runs, for the larger BHOSLIB benchmark instances. For each instance, the optimal MIS size is given by the two digits at the start of the instance name; 'S' gives the number of successful trials (from a total of 100) in which PLS located the optimal maximum independent set/minimum vertex cover; 'C' is the run-time in CPU seconds, averaged over all successful runs, for each instance. 'SAT' is the number of SAT solvers (from a total of 55 SAT solvers) in the SAT Competition 2004 that were able to solve the corresponding SAT problem (– indicates that the instance was not attempted).

FRB	MIS		MVC		SAT	FRB	MIS		MVC		SAT	FRB	MIS		MVC		SAT
	S	C	S	C			S	C	S	C			S	C	S	C	
35-17-1	100	3.6	100	4.36	–	45-21-3	100	363.21	100	353.34	–	53-24-5	94	917.18	94	931.83	–
35-17-2	100	1.25	100	1.80	–	45-21-4	100	53.95	100	48.70	–	56-25-1	18	1486.62	14	1396.17	0
35-17-3	100	0.22	100	0.25	–	45-21-5	100	80.49	100	127.05	–	56-25-2	9	1516.18	10	1360.71	0
35-17-4	100	4.29	100	6.12	–	50-23-1	84	910.49	87	827.32	1	56-25-3	12	1871.53	18	1773.34	–
35-17-5	100	0.52	100	1.08	–	50-23-2	55	992.89	59	1072.53	1	56-25-4	84	1376.57	89	1198.68	–
40-19-1	100	1.98	100	2.81	28	50-23-3	18	1198.42	22	1356.80	–	56-25-5	89	876.81	96	739.53	–
40-19-2	100	41.48	100	45.05	27	50-23-4	100	64.48	100	64.97	–	59-26-1	1	1040.82	2	1290.76	0
40-19-3	100	3.82	100	5.05	–	50-23-5	100	292.10	100	228.95	–	59-26-2	0	–	2	470.29	0
40-19-4	100	18.89	100	26.39	–	53-24-1	7	679.94	9	1537.44	0	59-26-3	10	1555.77	24	1612.06	–
40-19-5	100	81.68	100	118.69	–	53-24-2	29	1388.31	41	1609.19	0	59-26-4	5	1378.41	14	1729.49	–
45-21-1	100	29.49	100	33.89	8	53-24-3	79	1070.76	86	933.18	–	59-26-5	92	1094.49	97	814.93	–
45-21-2	100	63.5	100	81.29	5	53-24-4	49	1303.04	59	1203.43	–						

Table 2
PLS MIS and MVC performance results, averaged over 100 independent runs, for the more difficult DIMACS and DIMACS-C benchmark instances. 'S' gives the number of successful trials (from a total of 100) in which the optimal maximum independent set was located; 'α' is the putative optimal MIS size; 'β' is the putative optimal MVC size; 'C' is the run-time in CPU seconds, averaged over all successful runs, for each instance ('$<\epsilon$' signifies that the required CPU time is less than 0.01 s).

MIS							MVC						
Instance	DIMACS			DIMACS-C			Instance	DIMACS			DIMACS-C		
	S	α	C	S	α	C		S	β	C	S	β	C
brock800_1	100	10	0.02	100	23	15.38	brock400_1	100	393	$<\epsilon$	100	373	2.09
brock800_2	100	10	0.02	100	24	8.88	brock800_1	100	790	0.01	86	777	84.54
brock800_3	100	11	0.43	100	25	7.18	brock800_2	100	790	0.01	98	776	74.90
brock800_4	100	10	0.01	100	26	3.63	brock800_3	100	789	0.30	100	775	45.30
C2000.5	90	17	561.02	100	16	0.38	brock800_4	100	790	0.01	100	774	26.75
C2000.9	100	6	0.02	70	78	77.99	C1000.9	100	994	0.03	100	932	18.38
C4000.5	100	18	285.17	100	18	89.	C2000.5	86	1983	336.71	100	1984	0.50
keller6	100	63	0.08	100	59	84.69	C2000.9	100	1994	0.01	1	1922	36.28
MANN_a45	100	3	$<\epsilon$	30	344	276.43	C4000.5	100	3982	138.42	100	3982	142.02
MANN_a81	100	3	$<\epsilon$	0	1098	–	keller6	100	3298	0.03	95	3302	140.90
san1000	100	67	$<\epsilon$	100	15	16.55	MANN_a45	100	1032	$<\epsilon$	100	691	88.76
							MANN_a81	100	3318	$<\epsilon$	100	2223	485.79
							p_hat1500-1	100	1413	0.04	100	1488	1.39
							san1000	100	933	$<\epsilon$	100	985	19.81

putative maximum independent set sizes (α) for the DIMACS-C benchmark with the corresponding putative minimum vertex cover sizes $\beta = n - \alpha$ (where n is the number of vertices in G).

To evaluate the MIS/MVC performance of PLS, 100 independent PLS MIS and MVC trials were performed for each instance in the DIMACS and DIMACS-C benchmarks. The results from these experiments for the more difficult instances are displayed in Table 2. With regard to CPU requirements, the correlation between the PLS CPU times for finding MIS and MVC solutions for the DIMACS instances is 0.9967, and for the DIMACS-C instances, is 0.8405. With regard to the actual CPU times, it is noticeable that, for every instance in both the DIMACS and DIMACS-C benchmarks, solving the MVC problem consistently required more CPU time than solving the equivalent MIS problem. A significant reason for this could be that for all instances in the benchmarks, the maximum independent sets are considerably smaller than the minimum vertex covers and more PLS computational overhead is incurred in manipulating these larger sets.

Table 3 shows comparative DIMACS-C MIS results for PLS with the QSH [6] algorithm which clearly demonstrate that PLS achieves excellent performance on the DIMACS-C benchmark instances.

3.3. DIMACS-W and DIMACS-CW benchmark

The DIMACS benchmark instances were converted to weighted instances (DIMACS-W benchmark) by allocating weight, for vertex i, of $i \mod 200 + 1$ which allows future investigators to simply replicate the experiments performed in this study. The constant 200 in the weight calculation was determined after a number of experiments showed that the generated problems are reasonably difficult for PLS (clearly, allocating weights in the range $1, \ldots, k$ results in an MC instance when $k = 1$ while, intuitively, it is reasonable to expect that as k increases, the difficulty in solving the instance will, in general, increase). The DIMACS-CW benchmark consists of all the complement graphs of the DIMACS-W benchmark. For both the

Table 3
Comparative PLS MIS performance results for the DIMACS-C benchmark instances where either QSH or PLS required, on average, more than 1 s CPU time. The maximum independent set sizes found by the QSH [6] and PLS algorithms are shown in the correspondingly labeled α columns. The 'SC' column lists the scaled (to the reference computer used in this study) CPU time for the QSH algorithm and the 'C' gives the corresponding PLS CPU time (averaged over 100 trials). Entries of $< \epsilon$ signify that the average required CPU time is less than 0.01 s.

Instance	QSH		PLS		Instance	QSH		PLS		Instance	QSH		PLS	
	α	SC	α	C		α	SC	α	C		α	SC	α	C
brock400_1	27	2.33	27	0.52	c-fat500-10	126	2.22	126	$<\epsilon$	p_hat700-2	42	15.89	44	$<\epsilon$
brock400_2	29	2.33	29	0.21	c-fat500-2	26	3.44	26	$<\epsilon$	p_hat700-3	59	15.89	62	$<\epsilon$
brock400_3	31	2.22	31	0.10	c-fat500-5	64	2.56	64	$<\epsilon$	san400_0.5_1	9	2.22	13	0.13
brock400_4	33	2.33	33	0.05	johnson32-2-4	16	2.67	16	$<\epsilon$	san400_0.7_1	40	2.33	40	0.07
brock800_1	17	27.22	23	15.38	keller5	24	16.56	27	0.01	san400_0.7_2	30	2.11	30	0.09
brock800_2	24	27.00	24	8.88	p_hat500-1	9	5.33	9	$<\epsilon$	san400_0.7_3	16	2.33	22	0.09
brock800_3	25	25.33	25	7.18	p_hat500-2	33	5.44	36	$<\epsilon$	san400_0.9_1	100	2.56	100	$<\epsilon$
brock800_4	26	25.89	26	3.63	p_hat500-3	46	5.33	50	$<\epsilon$	sanr400_0.5	11	2.22	13	0.01
c-fat500-1	14	3.67	14	$<\epsilon$	p_hat700-1	8	15.89	11	0.01	sanr400_0.7	18	2.11	21	0.01

Table 4
PLS MWIS and MWVC performance for the DIMACS-W (DW) and DIMACS-CW (DCW) benchmark instances. 'W_s' and 'W_c' are the weights of the putative MWIS and MWVC found by PLS; '$\Delta'(= W_s + W_c - W_t)$ gives the relative error in the PLS results.

Instance	W_s DW	W_c DW	Δ	W_s DCW	W_c DCW	Δ	Instance	W_s DW	W_c DW	Δ	W_s DCW	W_c DCW	Δ
brock200_1	881	19 219	0	2 821	17 279	0	johnson32-2-4	4682	40 270	0	2 033	42 919	0
brock200_2	1538	18 562	0	1 428	18 672	0	johnson8-2-4	182	252	0	66	368	0
brock200_3	1213	18 887	0	2 062	18 038	0	johnson8-4-4	345	2 210	0	511	2 044	0
brock200_4	1132	18 968	0	2 107	17 993	0	keller4	2159	12 718	0	1 153	13 724	0
brock400_1	1057	39 143	0	3 422	36 778	0	keller5	5038	71 014	0	3 317	72 735	0
brock400_2	1039	39 161	0	3 350	36 850	0	keller6	9612	325 190	0	7 382	327 567	147
brock400_3	1072	39 128	0	3 471	36 729	0	MANN_a27	594	35 615	0	12 264	23 969	24
brock400_4	1068	39 132	0	3 626	36 574	0	MANN_a45	597	100 568	0	34 129	67 073	37
brock800_1	1573	78 827	0	3 121	77 279	0	MANN_a81	597	328 505	0	110 564	218 496	-42
brock800_2	1588	78 812	0	3 043	77 357	0	MANN_a9	135	945	0	372	708	0
brock800_3	1526	78 874	0	3 076	77 324	0	p_hat1000-1	9098	91 548	146	1 514	98 986	0
brock800_4	1530	78 870	0	2 971	77 429	0	p_hat1000-2	6815	93 685	0	5 777	94 723	0
C1000.9	855	99 645	0	8 965	91 738	203	p_hat1000-3	1569	98 931	0	7 986	92 593	79
C125.9	379	7 621	0	2 529	5 478	7	p_hat1500-1	9775	135 995	-80	1 619	144 231	0
C2000.5	2479	198 521	0	2 466	198 534	0	p_hat1500-2	7161	138 689	0	7 328	138 574	52
C2000.9	949	200 051	0	10 028	191 115	143	p_hat1500-3	1625	144 225	0	10 014	136 095	259
C250.9	597	20 828	0	5 092	16 333	0	p_hat300-1	4045	21 205	0	1 057	24 193	0
C4000.5	2776	399 228	4	2 792	399 215	7	p_hat300-2	2753	22 497	0	2 487	22 763	0
C500.9	705	44 645	0	6 822	38 593	65	p_hat300-3	1055	24 195	0	3 774	21 486	10
c-fat200-1	3294	16 806	0	1 284	18 816	0	p_hat500-1	5438	40 000	88	1 231	44 119	0
c-fat200-2	1728	18 372	0	2 411	17 689	0	p_hat500-2	4017	41 333	0	3 925	41 430	5
c-fat200-5	594	19 506	0	5 887	14 213	0	p_hat500-3	1281	44 069	0	5 361	40 027	38
c-fat500-1	6800	39 183	633	1 354	43 996	0	p_hat700-1	7026	58 479	55	1 441	64 009	0
c-fat500-10	788	44 562	120	11 586	33 764	0	p_hat700-2	5500	59 975	25	5 290	60 160	0
c-fat500-2	3500	41 970	0	2 628	42 722	0	p_hat700-3	1383	64 067	0	7 565	57 887	2
c-fat500-5	1544	43 806	0	5 841	39 509	0	san1000	7540	92 960	0	1 716	98 808	24
DSJC1000_5	2297	98 203	0	2 186	98 314	0	san200_0.7_1	1085	19 015	0	3 370	16 730	0
DSJC500_5	1876	43 474	0	1 725	43 625	0	san200_0.7_2	1473	18 627	0	2 422	17 678	0
gen200_p0.9_44	752	19 348	0	5 043	15 057	0	san200_0.9_1	590	19 510	0	6 825	13 275	0
gen200_p0.9_55	669	19 431	0	5 416	14 684	0	san200_0.9_2	699	19 401	0	6 082	14 018	0
gen400_p0.9_55	1073	39 127	0	6 718	33 665	183	san200_0.9_3	689	19 411	0	4 748	15 352	0
gen400_p0.9_65	987	39 213	0	6 935	33 265	0	san400_0.5_1	3754	36 446	0	1 455	38 745	0
gen400_p0.9_75	855	39 345	0	8 006	32 194	0	san400_0.7_1	1554	38 646	0	3 941	36 259	0
hamming10-2	398	100 426	0	50 512	50 312	0	san400_0.7_2	1891	38 309	0	3 110	37 090	0
hamming10-4	3006	97 818	0	5 086	95 707	-31	san400_0.7_3	2205	37 995	0	2 771	37 429	0
hamming6-2	129	2 015	0	1 072	1 072	0	san400_0.9_1	813	39 387	0	9 776	30 424	0
hamming6-4	650	1 494	0	134	2 010	0	sanr200_0.7	967	19 133	0	2 325	17 775	0
hamming8-2	398	21 354	0	10 976	10 776	0	sanr200_0.9	655	19 445	0	5 126	15 050	76
hamming8-4	2428	19 324	0	1 472	20 280	0	sanr400_0.5	1844	38 356	0	1 835	38 365	0
johnson16-2-4	1710	5 670	0	548	6 832	0	sanr400_0.7	1168	39 032	0	2 992	37 208	0

DIMACS-W and DIMACS-CW benchmarks, the putative maximum weighted independent sets/minimum weighted vertex covers were determined from a combination of long running PLS trials and shorter, more numerous trials.

When compared to MIS/MVC instances, the MWIS/MWVC instances have an extra degree of difficulty in that, while the optimal MWIS/MWVC solutions will be independent sets/vertex covers, they may not be maximum independent sets/minimum vertex covers. For the MWIS/MWVC performance of PLS on the DIMACS-W and DIMACS-CW benchmark instances, 100 independent trials were performed for each instance. The results from these experiments are displayed in

Table 4. For the DIMACS-W benchmark instances, the MIS/MVC correlation coefficient for CPU times is 0.6407 while that for the DIMACS-CW instances is 0.9057. Table 4 also identifies those instances where the relationship $W_s + W_c = W_t$ does not hold which signifies that, for these instances, PLS has not attained the optimal solution for either one or both of the MWIS and MWVC problems.

4. Conclusions and future work

This study has demonstrated that, by applying the general paradigm of dynamic local search to the maximum weight independent set problem, the state of the art in MIS/MVC/MWIS/MWVC solving can be improved. PLS is a stochastic reactive dynamic local search algorithm that interleaves sub-algorithms which alternate between sequences of iterative improvement, during which suitable vertices are added to the current sub-graph, and plateau search, during which vertices of the current sub-graph are swapped with vertices not contained in the current sub-graph. These sub-algorithms differ in firstly their vertex selection techniques in that selection can be solely based on randomly selecting a vertex, randomly selecting within highest/lowest vertex degree or randomly selecting within vertex penalties that are dynamically adjusted during the search. Secondly, the perturbation mechanism used to overcome search stagnation differs between the sub-algorithms. PLS has no problem instance dependent parameters and achieves state-of-the-art performance for the maximum weight independent set over a large range of the commonly used DIMACS and other benchmark instances.

The excellent performance of PLS on the benchmark instances reported here suggests that the underlying dynamic local search method has substantial potential to provide the basis for high-performance algorithms for other combinatorial optimisation problems.

References

[1] I. Bomze, M. Budinich, P. Pardalos, M. Pelillo, The maximum clique problem, in: D.-Z. Du, P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, vol. A., 1999, pp. 1–74.
[2] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness, Freeman, San Francisco, CA, USA, 1979.
[3] D. Johnson, M. Trick (Eds.), Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge, in: DIMACS Series, vol. 26, American Mathematical Society, 1996.
[4] L. Gibbons, D. Hearn, P. Pardalos, A continuous based heuristic for the maximum clique problem, in: D.S. Johnson, M.T. (Eds.), Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge, in: DIMACS Series, vol. 26, American Mathematical Society, 1996.
[5] C. Aggarwal, J. Orlin, R. Tai, Optimised crossover for the independent set problem, Operations Research 45 (1997) 226–234.
[6] S. Busygin, S. Butenko, P. Pardalos, A heuristic for the maximum independent set problem based on optimization of a quadratic over a sphere, Journal of Combinatorial Optimization 6 (2002) 287–297.
[7] V. Barbosa, L. Campos, A novel evolutionary formulation of the maximum independent set problem, Journal of Combinatorial Optimization 8 (2004) 419–437.
[8] W. Pullan, Phased local search for the maximum clique problem, Journal of Combinatorial Optimization 12 (2006) 303–323.