Fixed-Parameter Single Objective Search Heuristics for Minimum Vertex Cover

Wanru Gao[1], Tobias Friedrich[1,2], and Frank Neumann[1(✉)]

[1] School of Computer Science, The University of Adelaide, Adelaide, Australia
frank@cs.adelaide.edu.au
[2] Hasso Plattner Institute, Potsdam, Germany

Abstract. We consider how well-known branching approaches for the classical minimum vertex cover problem can be turned into randomized initialization strategies with provable performance guarantees and investigate them by experimental investigations. Furthermore, we show how these techniques can be built into local search components and analyze a basic local search variant that is similar to a state-of-the-art approach called NuMVC. Our experimental results for the two local search approaches show that making use of more complex branching strategies in the local search component can lead to better results on various benchmark graphs.

1 Introduction

The parameterized analysis of heuristic search methods has gained a lot of attention during the last few years [1,3,6–8,10]. It provides a mechanism for understanding how and why heuristic methods work for prominent combinatorial optimization problems. There are different methods closely related to the notion of fixed parameter algorithms. One popular paradigm to design parameterized algorithms are *bounded search tree algorithms* which search for a good solution by branching according to different rules that may be applied to solve the underlying problem.

For the classical vertex cover problem, different branching algorithms are available to answer the question whether a given graph has a vertex cover of size at most k. We investigate two common strategies resulting in fixed parameter algorithms running in time $\mathcal{O}^*(2^k)$ and $\mathcal{O}^*(\alpha^k)$[1], where $\alpha = 1.4656$, to solve this problem.

We show how these search tree algorithms can be turned into initialization approaches that produce initial solutions in linear time. We start by presenting an edge-based initialization approach which obtains a vertex cover having at most $k = 2\,OPT - r$, $0 \leq r \leq OPT$, nodes with probability at least $\binom{k}{OPT} \cdot 2^{-k}$.

[1] We use $\mathcal{O}^*(\cdot)$ to describe the essential functional behavior, ignoring all terms of lower order. For exponential expressions all polynomials are omitted: $\mathcal{O}^*(g(n)) = \mathcal{O}(g(n)\,\mathrm{poly}\,g(n))$.

J. Handl et al. (Eds.): PPSN XIV 2016, LNCS 9921, pp. 740–750, 2016.
DOI: 10.1007/978-3-319-45823-6_69

Furthermore, we present a node-based initialization approach which obtains an optimal solution with probability at least α^{-OPT}.

After having considered initialization approaches, we turn the branching rules into local search approaches and investigate their behaviour on different types of graphs. Both local search approaches start with a given vertex cover and try to find a smaller vertex cover by searching in the infeasible region of the search space. Our edge-based approach captures the essential ideas of a state-of-the-art local search algorithm for minimum vertex cover called NuMVC [2]. Having a vertex cover of size k, one node is removed to obtain a set of $k-1$. If this set is still a vertex cover, the algorithm searches for a vertex cover of size $k-2$ and so on. If the set is not yet a vertex cover an additional node is taken out and a node covering an uncovered edge is chosen. We turn these ideas in combination with our theoretical insights into an edge-based local search approach which obtains a vertex cover of size at most $k = 2\,OPT - r$ in an expected number of 2^{r+1} phases where each phase consists of a sequence of k local search steps.

Furthermore, we turn the node-based initialization approach into a similar local search approach and compare both local search strategies on different benchmark graphs. Our experimental results show that the node-based approach usually leads to a local search approach that obtains better solutions than the edge-based local search approach.

The paper is structured as follows. In Sect. 2, we provide some background material on parameterized algorithms and the minimum vertex cover problem. Section 3 introduces our two initialization heuristics and examines them from a theoretical and experimental perspective. Section 4 presents our two local search approaches and studies them on different types of benchmark instances. Finally, we provide some concluding remarks.

2 Preliminaries

The vertex cover problem is one of the best-known combinatorial optimization problems. Given an undirected graph $G = (V, E)$, the goal is to find a minimum set of vertices V' such that edge has at least one end vertex in V'. The problem is NP-hard and several 2-approximation algorithms are known. Furthermore, the problem has been studied extensively in the area of parameterized complexity. In fact, it is the archetypical problem in this area. Various kernelization approaches leading to fixed parameter algorithms of different runtime quality are known.

We make use of two branching approaches from the area of parameterized complexity [4]. Both have been introduced to determine whether a given graph $G = (V, E)$ contains a vertex cover of at most k nodes. The first approach builds on the fact that a vertex cover has to contain for each edge at least 1 node. It starts with G, picks an edge $e = \{u, v\}$ currently not covered, and branches according to the two options of including u or v. This allows to answer the question of whether G contains a vertex cover of size at most k in time $O^*(2^k)$.

The second approach makes more sophisticated decisions according to the degree of a node with respect to the uncovered edges. Considering a degree 1 node, it's always safe to take its neighbor. In the case of dealing with a degree 2

Algorithm 1. Edge-based Initialization Heuristic

```
1 $C := \emptyset$;
2 repeat
3     Let $e = \{u, v\}$ be a random uncovered edge, i.e., $e \in G[C]$;
4     with probability $1/2$ do
5         $C := C \cup \{u\}$
6     else
7         $C := C \cup \{v\}$
8 until $C$ is a vertex cover of $G$;
9 Return $C$;
```

node u, one has to choose either the two neighbors v and w of u or all neighbors (including u) of v and w. Finally, for a node u of degree at least 3, one has to choose u or all its neighbours. This approach allows to answer the question of whether G contains a vertex cover of size at most k in time $\mathcal{O}^*(\alpha^k)$, where $\alpha = 1.4656$.

We build on these two fixed parameter algorithms for the decision version of the vertex cover problem and study how to turn them into randomized initialization strategies with provable guarantees on their probability of achieving a solution of certain quality. In addition, we explore how they can be turned into local search approaches and study the performance of these approaches on benchmark instances.

For describing our algorithms we need one more piece of notation for each vertex cover $C \subseteq V$ of a graph $G = (V, E)$. We denote the subgraph of G consisting of the edges not covered by C and the corresponding non-isolated vertices by $G[C] := (V_C, E_C)$ with

$$E_C := E \setminus \{e \in E \mid e \cap C \neq \emptyset\} \text{ and}$$
$$V_C := \{v \in V \mid v \cap E_C \neq \emptyset\}.$$

Furthermore, we denote by $\deg_{G[C]}(u)$ the degree of a node u in $G[C]$ and by $N_{G[C]}[u]$ the set of neighbours of u in $G[C]$.

3 Initialization Strategies

We now describe two randomized initialization strategies based on the branching approaches described in the previous section. Both start with an empty set of nodes and add vertices until a vertex cover has been obtained. The edge-based initialization outlined in Algorithm 1 randomly selects in each step an uncovered edge and adds one of its endpoints chosen uniformly at random to the vertex cover.

For the edge-based initialization we can give a tradeoff between size of the obtained vertex cover and success probability.

Algorithm 2. Vertex-based Initialization Heuristic

```
$C := \emptyset$;
repeat
    if mindeg($G[C]$) = 1 then
        Let $u$ be a random node with $\deg_{G[C]}(u) = 1$;
        $C := C \cup N_{G[C]}[u]$ ;                                /* degree 1 rule */
    else
        Let $u$ be a node chosen uniformly at random from $G[C]$;
        if $\deg_{G[C]}(u) = 2$ then
            Let $v, w \in V$ such that $N_{G[C]}[u] = \{v, w\}$;
            with probability $\alpha^{-|N_{G[C]}[v] \cup N_{G[C]}[w]|}$ do
                $C := C \cup N_{G[C]}[v] \cup N_{G[C]}[w]$
            else
                $C := C \cup N_{G[C]}[u]$ ;                        /* degree 2 rule */
        else
            with probability $\alpha^{-\deg_{G[C]}(u)}$ do
                $C := C \cup N_{G[C]}[u]$
            else
                $C := C \cup \{u\}$ ;                              /* degree $\geq 3$ rule */
until $C$ is a vertex cover of $G$;
Return $C$;
```

Theorem 1. *For all r with $0 \leq r \leq OPT$, the edge-based initialization heuristic obtains a vertex cover of size at most $k := 2 \cdot OPT - r$ with probability at least $\binom{k}{OPT} \cdot 2^{-k}$.*

Proof. Let C^* be an optimal solution of value OPT. For each edge e at least one of its endpoints is contained in C^*. Hence, each step in the initialization process increases the number of nodes chosen from C^* by 1 with probability at least 1/2. We call a step increasing the number of nodes already chosen from C^* a *success*. OPT successes are sufficient to obtain a vertex cover. The probability to have OPT successes during k steps is at least $\binom{k}{OPT} \cdot 2^{-k}$. □

Observe that for $r := 0$ (and $k = 2\,OPT$), the edge-based initialization heuristic therefore obtains a 2-approximation of the minimum vertex cover with probability at least $\binom{2\,OPT}{OPT} \cdot 2^{-2\,OPT} = \Theta(1/\sqrt{OPT})$. On the other hand, for $r := OPT$ (and $k = OPT$), the edge-based initialization heuristic obtains a minimum vertex cover with probability at least 2^{-OPT}.

We now introduce an initialization heuristic based on more complex vertex-based branching. The vertex-based initialization given in Algorithm 2 first handles degree 1 nodes in the graph $G[C]$. If there is no degree 1 node in $G[C]$ then a node u in $G[C]$ is chosen uniformly at random and the degree rule for u is applied in a probabilistic way. To be more precise, if u is of degree 2 and v, w are its neighbours in $G[C]$ then all neighbours of v and w are added with

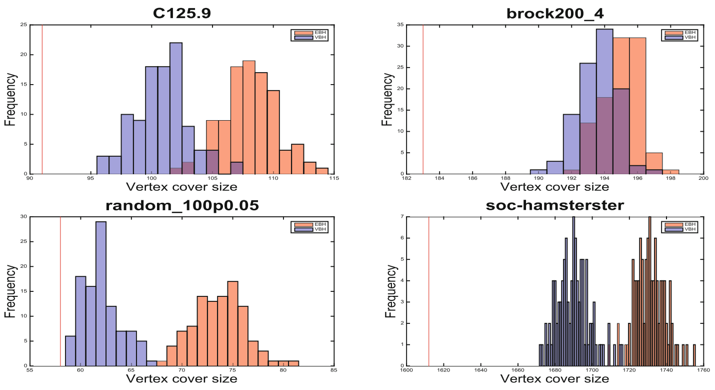

Fig. 1. The histograms show the frequency that each algorithm gets the initial vertex cover of certain size. The optimal vertex cover size of each instance is indicated with red vertical line in each figure.

probability $\alpha^{-|N_{G[C]}[v] \cup N_{G[C]}[w]|}$ while v and w are added otherwise. Similarly, if u is of degree at least 2 in $G[C]$ then all neighbours of u in $G[C]$ are added with probability at least $\alpha^{-\deg_{G[C]}(u)}$ while u is added otherwise.

We provide a lower bound on the probability that the vertex-based initialization obtains an optimal solution.

Theorem 2. *The vertex-based initialization heuristic obtains a vertex cover of size OPT with probability at least α^{-OPT}, where $\alpha = 1.4656$.*

Proof. The vertex-based initialization heuristics carries out a randomized branching according to the different rules. We distinguish the different cases regarding the degree of a node. For any graph, there is an optimal vertex cover that does not contain the node u if u is a degree one node. We investigate the degree 2 and 3 rules and show that each step i which requires selecting OPT_i nodes corresponding to an optimal solution occurs with probability at least α^{-OPT_i}. For a degree 2 node, there is an optimal vertex cover that contains either the neighbors v and w of u or all the neighbors of v and w. Note that a degree 2 rule is only applied if there is no node of degree 1 in $G[C]$. This implies that both v and w have to be connected to a node different from u. The probability of selecting v and w is $1 - \alpha^{-|N_{G[C]}[v] \cup N_{G[C]}[w]|}$ which is at least α^{-2} if $|N_{G[C]}[v] \cup N_{G[C]}[w]| \geq 2$. If $|N_{G[C]}[v] \cup N_{G[C]}[w]| = 1$, then v and w are connected and we have a cycle of length 3 $(u - v - w - u)$ for which selecting any subset of 2 nodes is optimal. Selecting u leads to an isolated edge $\{v, w\}$ for which the degree 1 rule selects a single vertex and therefore situations where $|N_{G[C]}[v] \cup N_{G[C]}[w]| = 1$ always lead to an optimal solution for the cycle of length 3. Finally, if u is of degree at least 3 there is an optimal vertex cover which either contains u or all the neighbors of u. The probability of selection u is $1 - \alpha^{-\deg_{G[C]}(u)} > \alpha^{-1}$.

Hence, the probability of selecting, in each step, a set of nodes leading to an optimal solution is at least

$$\prod_{i=1}^{\ell} \alpha^{-OPT_i} = \alpha^{-OPT}$$

where are ℓ is the number of iterations of the algorithm to produce the vertex cover. □

3.1 Experimental Investigations

In this section, we discuss about the experiments aiming at comparing the performance of Algorithms 1 and 2. Both algorithms are evaluated on sample Vertex Cover instances chosen from different benchmarks categories, which are *DIMACS* benchmarks, random generated undirected graphs and real world graphs.

There are some vertex cover benchmarks that are widely used to evaluated the performance of minimum vertex cover solver. One of these benchmarks is the *DIMACS* benchmark which is a set of challenge problems coming from the Second *DIMACS* Implementation Challenge for Maximum Clique, Graph Coloring and Satisfiability [5]. The original Max Clique problems from the challenge are converted to complement graphs and used as vertex cover problems. The random undirected graphs are generated with a pre-defined instance size and selection rate of edges. An edge between any two nodes is added to the graph with a certain pre-defined probability. In [9], there are a number of real world graphs with various number of vertices and edges. The sample graphs are selected from the undirected unweighted graphs.

Both of the algorithms are implemented in JAVA and the programs are executed for 101 independent repeated runs on each instance to obtain the statistics. The histograms in Fig. 1 are achieved by comparing the vertex cover sizes that the two algorithms get from running on four instances from different categories. The distribution of the solutions obtained in 101 independent runs is visualized with the histograms. In the first histogram and those lying in the second row, it is clear that vertex-based initialization generated smaller solutions for these two instances. For the instance brock200_4 from *DIMACS* benchmarks, the vertex-based approach has higher probability to generate better initial solutions than its edge-based counterpart.

Table 1 shows the five-number summary of each ranked set of 101 results testing on specific instance. From Table 1, the initial solutions of real world graphs generated by Algorithm 2 are all smaller than those from Algorithm 1. For the graphs from random and *DIMACS* benchmarks, the vertex-based approach can give better initial solutions for most times. Moreover, Algorithm 2 is able to generate solutions that are already global optimum for some of the instances in random and real world category.

Table 1. Experimental results on instances comparing the statistics between Algorithms 1 and 2.

Instance				EBH					VBH				
Name	$\|V\|$	$\|E\|$	OPT	min	Q1	Median	Q3	Max	Min	Q1	Median	Q3	Max
random_50p0.1	50	117	28	31	35	36	37	40	28	29	30	31	33
random_50p0.1-2	50	139	31	34	37	38	39	43	31	32	33	34	36
random_100p0.05	100	288	58	68	72	74	75	81	59	61	62	63	67
random_100p0.05-2	100	261	58	67	71	73	75	79	58	60	61	62	66
random_500p0.01	500	1 206	284	344	353	357	362	371	292	296	298	301	308
random_500p0.01-2	500	1 282	284	344	358	362	365	372	290	298	300	302	308
soc-hamsterster	2 426	16 630	1 612	1 709	1 726	1 731	1 737	1 755	1 672	1 684	1 690	1 695	1 716
soc-wiki-Vote	889	2 914	406	486	501	508	513	532	406	406	407	409	412
web-edu	3 031	6 474	1 451	1 742	1 765	1 771	1 780	1 793	1 451	1 452	1 453	1 454	1 457
web-google	1 299	2 773	498	582	596	604	611	632	501	506	508	509	517
bio-celegans	453	2 025	249	286	293	298	300	306	254	260	263	266	277
bio-yeast	1 458	1 948	456	583	608	618	626	656	456	459	460	462	468
brock200_4	200	6 811	183	192	194	195	196	198	190	193	194	194	197
brock400_4	400	20 035	367	390	392	393	394	396	387	390	391	392	395
brock800_4	800	111 957	774	792	794	795	796	798	792	793	794	794	797
C125.9	125	787	91	102	107	108	110	114	96	100	101	102	107
C250.9	250	3 141	206	227	231	232	234	238	222	225	226	228	232
C500.9	500	12 418	443	474	479	481	483	487	467	474	476	477	480

4 Local Search

We now introduce local search algorithms that make use of the aforementioned branching ideas. Both local search algorithms work with a list C representing a set of nodes and adding nodes to C in both algorithms always means adding them to the end of the list.

The edge-based local algorithm (see Algorithm 3) is a simplified version of one of the most successful approaches for solving the vertex cover problem, namely NuMVC [2]. It starts with a vertex cover of size $k+1$ and tries to find a smaller vertex cover of size k by removing one node. If this step violates the property of a vertex cover, it removes an additional node, picks an uncovered edge and adds one of its nodes uniformly at random. After a vertex cover of size k is obtained, it continues the process to search for a vertex cover of size $k-1$ and so on.

In the following, we give an upper bound on the number of steps of edge-based local search to find a vertex cover of size k. For our analysis, we partition the run of edge-based local search into distinct phases of length k which consist of k iterations of the while-loop.

Theorem 3. *For all r with $0 \leq r \leq OPT$, the edge-based local search finds a vertex cover of size $k := 2\,OPT - r$ after (expected) at most 2^{r+1} phases of length k.*

Proof. We investigate the probability that during k steps of the while-loop a vertex cover has been found at least once. We call this a *success* during a phase of k steps. Let C^* be a vertex cover of size OPT. As C^* is a vertex cover, it

Algorithm 3. Edge-based Local Search

```
1  Let $C$ be an initial vertex cover represented as a list;
2  repeat
3      Choose a node $v \in C$ uniformly at random and set $C := C \setminus v$;
4      while (($C$ is not a vertex cover of $G$) and (not termination condition)) do
5          Choose the first node $v$ of $C$ and set $C := C \setminus v$;
6          Let $e = \{u, v\}$ be a random uncovered edge, i.e., $e \in G[C]$;
7          with probability $1/2$ do
8              $C := C \cup \{u\}$
9          else
10             $C := C \cup \{v\}$
11 until termination condition;
12 Return $C$;
```

contains for each edge $e \in E$ at least one vertex. Consider an edge $e = \{u, v\}$. At each iteration, a vertex $z \in C^*$ is picked with probability at least $1/2$ and each node of C^* is picked at most once as only uncovered edges are chosen. The expected number of distinct vertices contained in C^* during a phase of k steps is therefore at least $k/2 = (2\,OPT - r)/2$. The probability that during the first r steps only nodes of C^* are picked is at least 2^{-r}. The expected number of nodes of C^* picked in the remaining $2\,OPT - 2r$ steps (before a vertex cover is reached) is at least $OPT - r$. Furthermore, it is at least $OPT - r$ with probability $1/2$. Hence, the algorithm picks all OPT nodes during a phase of $k = 2\,OPT - r$ steps with probability at least $2^{-(r+1)}$. The expected number of phases of length k needed to find a vertex cover is therefore at most 2^{r+1}. □

We also turn the vertex-based branching approach into a vertex-based local search algorithm (see Algorithm 4). This approach searches for a vertex cover after removing a node together with all its neighbors. Afterwards, it tries to obtain a new vertex cover by picking a random node of minimum degree in the graph consisting of currently all uncovered edges. Based on the degree of this node the degree rules are applied with the already introduced biased probabilities. The last step is iterated until a vertex cover is found again.

4.1 Experimental Investigations

We test Algorithms 3 and 4 on some sample instances to evaluate their performance. Both algorithms are given an initial vertex cover produced by Algorithm 1 and the cut off generation is set to 100 000. Both algorithms are implemented in JAVA and their performance is measured by the number of iterations it takes to make improvement.

Algorithm 4. Vertex-based Local Search

```
Set $\alpha := 1.4656$;
Let $C$ be an initial vertex cover represented as a list;
repeat
    Choose the first node $v$ of $C$ and set $C := C \setminus N_G^2[v]$;
    repeat
        Let $u$ be a random node with $\deg_{G[C]}(u) = \text{mindeg}(G[C])$;
        if $\deg_{G[C]}(u) = 1$ then
            $C := C \cup N_{G[C]}[u]$ ;                          /* degree 1 rule */
        else if $\deg_{G[C]}(u) = 2$ then
            Let $v, w \in V$ such that $N_{G[C]}[u] = \{v, w\}$;
            with probability $\alpha^{-|N_{G[C]}[v] \cup N_{G[C]}[w]|}$ do
                $C := C \cup N_{G[C]}[v] \cup N_{G[C]}[w]$
            else
                $C := C \cup N_{G[C]}[u]$ ;                      /* degree 2 rule */
        else
            with probability $\alpha^{-\deg_{G[C]}(u)}$ do
                $C := C \cup N_{G[C]}[u]$
            else
                $C := C \cup \{u\}$ ;                            /* degree $\geq 3$ rule */
    until $C$ is a vertex cover of $G$ (or termination condition);
until termination condition;
Return $C$;
```

Figure 2 shows the improvement of the two algorithms on example instances over iterations. $|C| - OPT$ denotes the size difference between the best solution so far and the globally optimal solution. The stairstep lines are drawn for three independent runs for each instance and algorithm. The vertex-based heuristic makes significant improvement before 2 000 generations for these three instances from the observation of the solid lines while the solution of edge-based heuristic does not improve much until 100 000 which is the cutoff bound. For the random graphs, the vertex-based approach is able to find a global optimum before 10 000 iterations whereas the edge-based heuristic does not reach the optimal solution before 100 000 iterations.

More results are shown in Table 2. The average best vertex cover sizes at certain number of iterations from 10 independent runs of these two algorithms on a certain vertex cover problem are listed in the table. From the statistics in Table 2, vertex-based approach produces better results for 15, 15, 16 and 16 out of the 17 instances after 10 000, 50 000, 100 000 and 200 000 iterations, respectively. Moreover, Algorithm 4 has a success rate of 100 % in solving 8 instances from different categories.

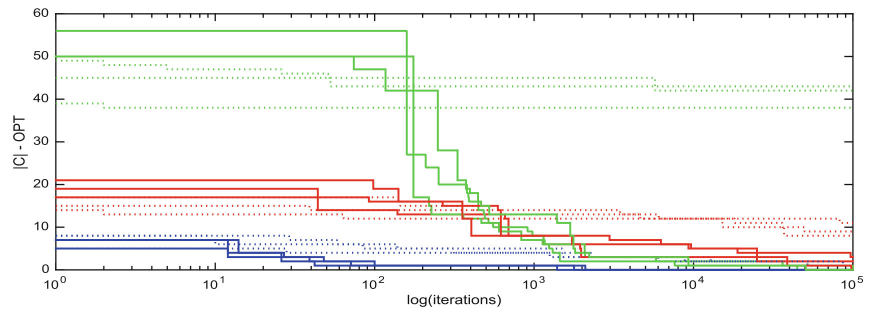

Fig. 2. The improvement of both algorithms in three example instances over iterations. The lines in blue, red and green color represent an independent run on the instance random-50prob10, C125.9 and bio-celegans, respectively. The dotted lines and solid lines denote the results from Algorithms 3 and 4.

Table 2. Performance comparison between Algorithms 3 and 4 on some sample instances. The average vertex cover size is listed after running each algorithm for certain number of iterations.

Instance				EBH				VBH			
Name	$\|V\|$	$\|E\|$	OPT	10 000	50 000	100 000	200 000	10 000	50 000	100 000	200 000
random_50p0.1	50	117	28	29.8	29.4	28.9	28.8	28.0	28.0	28.0	28.0
random_50p0.1-2	50	139	31	33.0	32.6	32.1	32.0	31.0	31.0	31.0	31.0
random_100p0.05	100	288	58	66.7	65.4	65.1	64.8	58.0	58.0	58.0	58.0
random_100p0.05-2	100	261	58	66.4	64.8	64.3	64.0	58.0	58.0	58.0	58.0
random_500p0.01	500	1 206	284	351.3	348.8	348.2	346.6	286.4	284.9	284.4	284.4
random_500p0.01-2	500	1 282	284	357.0	354.9	353.1	352.3	286.3	284.2	284.2	284.0
bio-celegans	453	2 025	249	291.4	290.7	290.0	289.8	250.7	249.7	249.3	249.3
bio-diseasome	516	1 188	285	316.2	314.5	314.5	313.3	288.9	287.3	287.0	286.6
soc-dolphins	62	159	34	36.3	35.7	35.4	34.9	34.0	34.0	34.0	34.0
soc-wiki-Vote	889	2 914	406	502.2	502.2	502.2	502.2	406.2	406.0	406.0	406.0
ca-netscience	379	914	214	243.9	241.1	240.3	238.7	216.7	215.7	215.1	214.7
ca-Erdos992	6 100	7 515	461	819.1	808.3	801.2	794.9	461.0	461.0	461.0	461.0
C125.9	125	787	91	102.7	101.1	100.5	100.4	95.3	93.3	92.8	92.8
C250.9	250	3 141	206	228.2	226.8	226.3	225.6	232.5	231.5	231.2	230.6
MANN_a27	378	702	252	261.0	260.8	260.4	260.1	252.9	252.6	252.3	252.1
MANN_a45	1 035	1 980	690	705.0	705.0	705.0	705.0	701.6	694.2	693.3	692.7
MANN_a81	3 321	6 480	2 221	2 241.0	2 241.0	2 241.0	2 241.0	2 241.4	2 241.1	2 239.0	2 235.1

5 Conclusions

We have shown how well-known fixed parameter branching algorithms for the minimum vertex cover problem can be turned into randomized initialization strategies and guarantee the probabilities of obtaining good solutions. Furthermore, we have incorporated the branching rules into local search algorithms and observed that the edge-based local search algorithm is equivalent to the core component of the state-of-the-art local search algorithm called NuMVC. Considering the edge-based local search algorithm from a theoretical perspective we have shown fixed parameter and trade-off results on its performance. Additionally, we have demonstrated how the more complex vertex-based branching rules can be incorporated into the vertex-based local search algorithm and shown that this usually leads to better results on random graphs and social networks than edge-based local search.

Acknowledgements. This research has been supported by the Australian Research Council (ARC) under grant agreement DP140103400 and the German Science Foundation (DFG) under grant agreement no. FR 2988.

References

1. Bringmann, K., Friedrich, T.: Parameterized average-case complexity of the hypervolume indicator. In: Genetic and Evolutionary Computation Conference (GECCO), pp. 575–582 (2013)
2. Cai, S., Su, K., Sattar, A.: Two new local search strategies for minimum vertex cover. In: Twenty-Sixth AAAI Conference on Artificial Intelligence (2012)
3. Corus, D., Lehre, P.K., Neumann, F., Pourhassan, M.: A parameterised complexity analysis of bi-level optimisation with evolutionary algorithms. Evol. Comput. **24**, 183–203 (2015)
4. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer, Heidelberg (2013)
5. Johnson, D.J., Trick, M.A. (eds.): Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11–13, 1993. American Mathematical Society, Boston (1996)
6. Kratsch, S., Neumann, F.: Fixed-parameter evolutionary algorithms and the vertex cover problem. Algorithmica **65**, 754–771 (2013)
7. Kratsch, S., Lehre, P.K., Neumann, F., Oliveto, P.S.: Fixed parameter evolutionary algorithms and maximum leaf spanning trees: a matter of mutation. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 204–213. Springer, Heidelberg (2010)
8. Nallaperuma, S., Sutton, A.M., Neumann, F.: Parameterized complexity analysis and more effective construction methods for ACO algorithms and the euclidean traveling salesperson problem. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 2045–2052. IEEE (2013)
9. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI, pp. 4292–4293 (2015)
10. Sutton, A.M., Neumann, F., Nallaperuma, S.: Parameterized runtime analyses of evolutionary algorithms for the planar euclidean traveling salesperson problem. Evol. Comput. **22**, 595–628 (2014)