Separate, Measure and Conquer: Faster Polynomial-Space Algorithms for Max 2-CSP and Counting Dominating Sets

Serge Gaspers[1(✉)] and Gregory B. Sorkin[2]

[1] UNSW Australia and NICTA, Sydney, Australia
sergeg@cse.unsw.edu.au
[2] London School of Economics, London, UK
g.b.sorkin@lse.ac.uk

Abstract. We show a method resulting in the improvement of several polynomial-space, exponential-time algorithms. The method capitalizes on the existence of small balanced separators for sparse graphs, which can be exploited for branching to disconnect an instance into independent components. For this algorithm design paradigm, the challenge to date has been to obtain improvements in worst-case analyses of algorithms, compared with algorithms that are analyzed with advanced methods, such as Measure and Conquer. Our contribution is the design of a general method to integrate the advantage from the separator-branching into Measure and Conquer, for an improved running time analysis.

We illustrate the method with improved algorithms for Max $(r, 2)$-CSP and #Dominating Set. For Max $(r, 2)$-CSP instances with domain size r and m constraints, the running time improves from $r^{m/6}$ to $r^{m/7.5}$ for cubic instances and from $r^{0.19 \cdot m}$ to $r^{0.18 \cdot m}$ for general instances, omitting subexponential factors. For #Dominating Set instances with n vertices, the running time improves from 1.4143^n to 1.2458^n for cubic instances and from 1.5673^n to 1.5183^n for general instances. It is likely that other algorithms relying on local transformations can be improved using our method, which exploits a non-local property of graphs.

1 Introduction

Graph separators have been used for divide-and-conquer algorithms since the 70s [21]. For classes of instances with sublinear separators, e.g., planar graphs, this often gives subexponential- or polynomial-time algorithms. It is natural to design a branching strategy that strives to disconnect an instance into components, even when no sublinear separators are known. While this has successfully been done experimentally [3,4,11,17,20], we are not aware of worst-case analyses of branching algorithms that are based on linear separators. Our algorithms exploit small separators, specifically, balanced separators of size about $n/6$ for cubic graphs of order n. Their existence is known since 2001, they have been used in pathwidth-based algorithms using exponential time and exponential space, and it is natural to try to exploit them for polynomial-space algorithms.

We now introduce the main separation results. We use standard graph notation from [5]. In a graph $G = (V, E)$, the *contraction* of an edge $uv \in E$ is an operation replacing u and v by one new vertex c_{uv} that is adjacent to $N_G(\{u, v\})$. The graph G is *cubic* or 3-*regular* if each vertex has degree 3 and *subcubic* if each vertex has degree at most 3.

Let (L, S, R) be a partition of the vertex set of a graph G such that there is no edge in G with one endpoint in L and the other endpoint in R. We say that (L, S, R) is a *separation* of G, and that S is a *separator* of G, *separating* L and R. The following lemma follows from results in [9,22].

Lemma 1. *For any subcubic graph G with n vertices, a separation (L, S, R) with $|S| \leq \frac{n}{6} + o(n)$ and $|L|, |R| \leq \frac{n-|S|+1}{2}$ can be computed in polynomial time.*

A direct application of branching on the vertices in such a separator yields algorithms inferior to existing Measure and Conquer ones. Our improvements have their origin in a simple observation: if an algorithm can always branch on vertices in the separator, then the usual measure of improvement is achieved at each step, and the splitting of the graph into two parts when the separator is emptied is a bonus. We get the best of both. The technical challenges are to amortize this bonus over the previous branches to prove a better running time, and to control the balance of the separation as the algorithm proceeds so that the bonus is significant.

We illustrate for cubic Max 2-CSP. We will be optimistic in this sketch, doing the analysis rigorously in Section 2. The problem class will also be defined there, but for now one may think of Max Cut, with domain size $r = 2$. Let us "pivot" on a vertex $v \in S$, i.e., sequentially assign it each possible value, eliminate it and its incident edges (see rule R3 below), and solve each case recursively. It is possible that v has neighbors within S, but this is a favorable case, reducing the number of subsequent branches needed. So, suppose that v has neighbors only in L and R. If all neighbors were in one part, the separator could be made smaller, so let us skip over this case as well. The cases of interest, then, are when v has two neighbors in L and one in R, or vice-versa. Suppose that these cases occur equally often; this is the bit of optimism that will require more care to get right. In that case, after all $|S|$ branchings, the sizes of L and R are each reduced by $\frac{3}{2}|S|$, since degree-2 vertices get contracted away. This would lead to a running time bound $t(n)$ satisfying the recurrence

$$t(n) = r^{n/6} \cdot 2t(\tfrac{5}{12}n - \tfrac{3}{2} \cdot \tfrac{1}{6}n),$$

leading to a solution with $t(n) = O^\star(r^{n/5})$. This conjectured bound would improve on the best previous time bound of $O^\star(r^{n/4})$, and Section 2 establishes that the bound is true, modulo a subexponential factor in the running time.

Our algorithms exploit a *global* graph structure, the separator, while executing an algorithm based on *local* simplification and branching rules. The use of global structure may also make it possible to circumvent lower bounds for classes of algorithms restricted to local information [1,2].

Results. Section 2 gives a first analysis of a separator-based algorithm. It solves cubic instances of Max 2-CSP in time $r^{(1/5+o(1))\,n}$, where n is the number of vertices, improving on the previously fastest $O^\star(r^{n/4})$ time polynomial-space algorithm [25]. By [25, Theorem22], this cubic result allows solution of general instances of Max 2-CSP in time $r^{(9/50+o(1))\,m}$, improving on the previously fastest $O^\star(r^{(19/100)\,m})$ polynomial-space algorithm [25]. The latter improvement holds also for Max Cut, an important special case of Max 2-CSP, and for Polynomial and Ring CSP, generalizations encompassing graph bisection, the Ising model, and counting problems (see [26]).

While Max 2-CSP is a central problem in exponential-time algorithms, the analysis of its branching algorithms is typically easier than for other problems, largely because the branching creates isomorphic subinstances. In Section 3, we develop the Separate, Measure and Conquer method in full generality, and use this in Section 4 to design faster polynomial-space algorithms for counting dominating sets. For graphs with maximum degree 3, we obtain an algorithm with a time bound of $3^{(1/5+o(1))\,n} = O(1.2458^n)$, improving on the previous best $O^\star(2^{(1/2)\,n}) = O(1.4143^n)$ [19]. For general graphs, we obtain a different algorithm, with time bound $O(1.5183^n)$, improving on the previous best $O(1.5673^n)$ [27]. For details and proofs omitted from this conference version, see [14].

2 Max 2-CSP

Using the notation from [25], an instance (G, S) of Max 2-CSP (also called Max $(r, 2)$-CSP) is given by a *constraint graph* $G = (V, E)$ and a set S of *score functions*. Writing $[r] = \{1, ..., r\}$ for the set of available vertex colors, we have a *dyadic* score function $s_e : [r]^2 \to \mathbb{R}$ for each edge $e \in E$, a *monadic* score function $s_v : [r] \to \mathbb{R}$ for each vertex $v \in V$, and a single *niladic* score "function" $s_\emptyset : [r]^0 \to \mathbb{R}$ which is just a constant convenient for bookkeeping. A *candidate solution* is a function $\phi : V \to [r]$ assigning colors to the vertices (ϕ is an *assignment* or *coloring*), and its score is

$$s(\phi) := s_\emptyset + \sum_{v \in V} s_v(\phi(v)) + \sum_{uv \in E} s_{uv}(\phi(u), \phi(v)).$$

An *optimal solution* ϕ is one which maximizes $s(\phi)$.

Let us recall the reductions from [25]. R0–R2 are simplification rules, creating one subinstance, and R3 is a branching rule, creating r subinstances. An optimal solution for (G, S) can be found in polynomial time from optimal solutions of the subinstances.

R0 If $d(y) = 0$, then set $s_\emptyset = s_\emptyset + \max_{C \in [r]} s_y(C)$ and delete y from G.
R1 If $N(y) = \{x\}$, then replace the instance with (G', S') where $G' = (V', E') = G - y$ and S' is the restriction of S to V' and E' except that for all $C \in [r]$ we set

$$s'_x(C) = s_x(C) + \max_{D \in [r]} \{s_{xy}(C, D) + s_y(D)\}.$$

R2 If $N(y) = \{x, z\}$, then replace the instance with (G', S') where $G' = (V', E') = (V - y, (E \setminus \{xy, yz\}) \cup \{xz\})$ and S' is the restriction of S to V' and E', except that for $C, D \in [r]$ we set

$$s'_{xz}(C, D) = s_{xz}(C, D) + \max_{F \in [r]}\{s_{xy}(C, F) + s_{yz}(F, D) + s_y(F)\}$$

if there was already an edge xz, discarding the first term $s_{xz}(C, D)$ otherwise.

R3 Let y be a vertex of degree at least 3. There is one subinstance (G', s^C) for each color $C \in [r]$, where $G' = (V', E') = G - y$ and s^C is the restriction of s to V' and E', except that we set

$$(s^C)_\emptyset = s_\emptyset + s_y(C), \quad \text{and} \quad (s^C)_x(D) = s_x(D) + s_{xy}(D, C)$$

for every neighbor x of y and every $D \in [r]$.

We will now describe a new separator-based algorithm for cubic Max 2-CSP, outperforming the algorithm from [25]. Using it as a subroutine in the algorithm for general instances [25] also gives a faster running time for Max 2-CSP.

2.1 Background

For a cubic instance of Max 2-CSP, an instance whose constraint graph G is 3-regular, the fastest known polynomial-space algorithm makes simple use of the reductions above. The algorithm branches on a vertex v of degree 3, giving r instances with a common constraint graph G', where v has been deleted. In G', the three G-neighbors of v each have degree 2. Simplification rules are applied to rid G' of degree-2 vertices, and further vertices of degree 0, 1, or 2 that may result, until the constraint graph becomes another cubic graph G''. This results in r instances with the common constraint graph G'', to which the same algorithm is applied recursively. The running time of the algorithm is exponential in the number of branchings, and since each branching destroys 4 degree-3 vertices (the pivot vertex v and its three neighbors), the running time is bounded by $O^\star(r^{n/4})$; details may be found in [24].

Here, we break this $r^{n/4}$ barrier by selecting pivot vertices using global properties of the graph. Our algorithm pivots only on vertices in a separator; when the separator is exhausted, G has been split into components L and R which can be solved independently. The efficiency gain comes from the component splitting: if the time to solve an instance with n vertices is $O^\star(r^{cn})$, the time to solve an instance consisting of components L and R is $O^\star(r^{c|L|}) + O^\star(r^{c|R|})$, which (for L and R of comparable sizes) is hugely less than the time bound $O^\star(r^{c(|L|+|R|)})$ for a single component of the same total order. This efficiency gain comes at no cost: until the separator is exhausted, branching on vertices in the separator is just as efficient as branching on any other vertex.

2.2 Analysis

To analyze the algorithm, we use the Measure and Conquer method. Our measure associates a non-negative real to each instance. As in [13], we use penalty terms in the measure to treat tricky cases. We also take from [25] and [13] the treatment of vertices of degrees 1 and 2 within the Measure and Conquer framework.

Recall [8,12,13] that the Measure and Conquer analysis applies to an algorithm which polynomially transforms an instance I to one or more instances $I_1, \dots, I_k$, solves those instances recursively, and obtains a solution to I in polynomial time from the solutions of $I_1, \dots, I_k$. The measure $\mu(I)$ of an instance I should satisfy that for any instance,

$$\mu(I) \geq 0, \tag{1}$$

and for any transformation of I into $I_1, \dots, I_k$,

$$r^{\mu(I_1)} + \cdots + r^{\mu(I_k)} \leq r^{\mu(I)}. \tag{2}$$

Given these hypotheses, the algorithm solves any instance I in time $O^\star(r^{\mu(I)})$ if the number of recursive calls from the root to a leaf of the search tree is polynomial.

Here, we present an instance of Max 2-CSP in terms of a separation (L, S, R) of its constraint graph $G = (V, E)$. We write L_3, S_3, and R_3 for the subsets of degree-3 vertices of L, S, and R, respectively, and we will always assume that $|L_3| \leq |R_3|$, if necessary swapping the roles of L and R to make it so. We write $|S_2|$ for the number of degree-2 vertices in S. We define the measure of an instance as

$$\begin{aligned}\mu(L, S, R) = w_s|S_3| + w_{s,2}|S_2| + w_r|R_3| + w_b\mathbb{1}(|R_3| = |L_3|) \\ + w_c\mathbb{1}(|R_3| = |L_3| + 1) + w_d \log_{3/2}(|R_3| + |S_3|),\end{aligned} \tag{3}$$

where the values w_s, $w_{s,2}$, w_r, w_b, w_c, and w_d are constants to be determined and the indicator function $\mathbb{1}(\text{event})$ takes the value 1 if the event is true and 0 otherwise. For the constraint (1) that $\mu \geq 0$, it suffices to constrain each of the constants to be nonnegative:

$$w_s, w_{s,2}, w_r, w_b, w_c, w_d \geq 0. \tag{4}$$

Intuitively, the terms w_b and w_c are the only representations of the size of L in μ, and account for the greater time needed when the left side is as large (or nearly as large) as the right. The logarithmic term offsets increases in penalty terms that may result when a new separator is computed, where the instance may go from imbalanced to balanced.

Concretely, from (2), each reduction imposes a constraint on the measure. We treat the reductions in their order of priority: when presenting one reduction, we assume that no previous reduction can be applied. Denote by μ the value of the measure before the reduction is applied and by μ' its value after the reduction.

Degree 0. If the instance contains a vertex v of degree 0, then perform R0 on v. Removing v has no effect on the measure and Condition (2) is satisfied.

Half-edge deletion. A half-edge deletion occurs when the degree of a vertex v decreases. We will require that a degree decrease does not increase the measure, which will validate R1, the collapse of parallel edges, and R2 for vertices in $L \cup R$. If $d(v) \leq 2$, Condition (2) is satisfied since $w_{s,2} \geq 0$ by (4), and v only affects the measure if $v \in S$ and $d(v) = 2$. Now, assume $d(v) = 3$. Taking into account changes in imbalance, and separately analyzing the cases where v is in L, S, and R, we obtain the constraints:

$$-w_b + w_c \leq 0 \qquad (5) \qquad\qquad -w_r + w_c \leq 0 \qquad (7)$$

$$-w_s + w_{s,2} \leq 0 \qquad (6) \qquad \text{and} \quad -w_r + w_b - w_c \leq 0. \qquad (8)$$

Separation. This reduction is the only one special to separation, and its constraint looks quite different from those in previous works. The reduction applies when $S = \emptyset$, which arises in two cases. One is at the beginning of the algorithm, when the instance has not been separated, and may be represented by the trivial separation $(\emptyset, \emptyset, V)$. The second is when reductions on separated instances have exhausted the separator, so that S is empty but L and R are nonempty, and the instance is solved by solving the instances on L and R independently, via a new separation (L', S', R') for R and another such separation (L'', S'', R'') for L. The reduction is applied to a graph $G = (V, E)$ that is cubic and can be assumed to be of at least some constant order, $|V| \geq k$, since a smaller instance can be solved in constant time. By Lemma 1 we know that, for any constant $\epsilon > 0$, there is a size $k = k(\epsilon)$ such that any cubic graph G of order at least k has a separation (L, S, R) with $|S| \leq (\frac{1}{6} + \epsilon)|V|$, $|S|, |R| \leq \frac{5}{12}|V|$. From (2), making worst-case assumptions about balance, it suffices to constrain that

$$r^{w_s|S_3'|+w_r|R_3'|+w_b+w_d \log(|R_3'|+|S_3'|)} + r^{w_s|S_3''|+w_r|R_3''|+w_b+w_d \log(|R_3''|+|S_3''|)} \leq r^{w_r|R_3|+w_d \log(|R_3|)}.$$

From the separator properties, this in turn is implied by

$$2 \cdot r^{w_s(1/6+\epsilon)|R_3|+w_r(5/12 \cdot |R_3|)+w_b+w_d \log(8/12 \cdot |R_3|)} \leq r^{w_r|R_3|+w_d \log(|R_3|)},$$

where we have estimated $|L_3'|, |R_3'| \leq \frac{5}{12}|R_3|$ and $|S_3'| \leq (\frac{1}{6} + \epsilon)|R_3| \leq \frac{3}{12}|R_3|$ in the log term on the left hand side. Since $r \geq 2$, it suffices to constrain that

$$1 + w_s(\tfrac{1}{6} + \epsilon)|R_3| + w_r(\tfrac{5}{12}|R_3|) + w_b + w_d \log(\tfrac{8}{12}|R_3|) \leq w_r|R_3| + w_d \log(|R_3|).$$

Taking $\frac{3}{2} = \frac{12}{8}$ to be the logarithm's base and setting $w_d = w_b + 1$, the left term $w_d \log(\frac{8}{12}|R_3|)$ is equal to $-(w_b + 1) + (w_b + 1)\log(|R_3|)$, and it suffices to have

$$(\tfrac{1}{6} + \epsilon)w_s + \tfrac{5}{12}w_r \leq w_r. \qquad (9)$$

Degree 2 in S. If the instance has a vertex $s \in S$ of degree 2, then perform R2 on s. Let $N(s) = \{u_1, u_2\}$. The vertex s is removed and the edge u_1u_2 is added if it was not present already. If L or R contain no neighbor of s, Condition (2) is implied by the constraints of the half-edge deletions. If $u_1 \in L$ and $u_2 \in R$ (or the symmetric case), then S is not a separator any more. The algorithm removes u_2 from R and adds it to S. If $d(u_2) = 2$, we have that $\mu' - \mu \leq 0$. Otherwise, $d(u_2) = 3$ and $\mu' - \mu \leq -w_{s,2} + w_s - w_r + \max(0, w_c, w_b - w_c)$. Since $w_c \geq 0$ by (4) it suffices to constrain

$$-w_{s,2} + w_s - w_r + w_c \leq 0 \quad (10) \quad \text{and} \quad -w_{s,2} + w_s - w_r + w_b - w_c \leq 0. \quad (11)$$

No neighbor in L. If the separation (L, S, R) has a vertex $v \in S$ with no neighbor in L, "drag" v into R, i.e., transform the instance by changing the separation to $(L', S', R') := (L, S \setminus \{v\}, R \cup \{v\})$. It is easily checked that this is a valid separation, and with $|L_3'| \leq |R_3'|$ implied by $|L_3| \leq |R_3|$. Indeed the new instance is no more balanced than the old, so that the difference between the new and old measures is $\mu' - \mu \leq -w_s + w_r$, and to satisfy condition (2) it suffices that

$$-w_s + w_r \leq 0, \tag{12}$$

since by (4) and (5) an increase in imbalance does not increase the measure.

No neighbor in R. A vertex $v \in S$ with no neighbor in R is dragged into L. The case where $|R_3| = |L_3|$ is covered by the previous case, reversing the roles of L and R. Otherwise, $|R_3| \geq |L_3| + 1$, and $\mu' - \mu \leq -w_s + \max(0, w_c, w_b - w_c)$. We constrain that

$$-w_s + w_c \leq 0 \quad (13) \quad \text{and} \quad -w_s + w_b - w_c \leq 0. \quad (14)$$

With the above cases covered, we may assume that the pivot vertex $s \in S$ has degree 3 and at least one neighbor in each of L and R.

One neighbor in each of L, S, and R. To branch on a vertex $s \in S$ with one neighbor in each of L, S, and R, perform R3 on s, deleting it from the constraint graph. Since both L and R lose a degree-3 vertex, there is no change in balance and the constraint is

$$1 - 2w_s + w_{s,2} - w_r \leq 0. \tag{15}$$

The form and the initial 1 come from the reduction's generating r instances with common measure μ', so the constraint is $r \cdot r^{\mu'} \leq r^{\mu}$, or equivalently $1 + \mu' - \mu \leq 0$. The value of $\mu' - \mu$ comes from S losing two degree-3 vertices but gaining a degree-2 vertex, and R losing a degree-3 vertex.

Two neighbors in L. If $s \in S$ has two neighbors in L and one neighbor in R, applying R3 removes s, reduces the degree of a degree-3 vertex in R, and increases the imbalance by one. The algorithm performs R3 if $|R_3| \leq |L_3| + 1$, where $\mu' - \mu \leq -w_s - w_r + \max(-w_b + w_c, -w_c)$. Thus, we constrain

$$1 - w_s - w_r - w_b + w_c \leq 0 \quad (16) \quad \text{and} \quad 1 - w_s - w_r - w_c \leq 0. \quad (17)$$

If, instead, $|R_3| \geq |L_3| + 2$, then the algorithm drags s into L and its neighbor $r \in R$ into S, replacing (L, S, R) by $(L \cup \{s\}, (S \setminus \{s\}) \cup \{r\}, R \setminus \{r\})$. We need to ensure that $-w_r + \max(w_b, w_c) \leq 0$, which, since $w_c \leq w_b$ by (5), is satisfied if we constrain

$$-w_r + w_b \leq 0. \tag{18}$$

Two neighbors in R***.*** If $s \in S$ has two neighbors in R and one neighbor in L, the algorithm performs R3, which removes s, reduces the degree of two degree-3 vertices in R, and decreases the imbalance by one. For the case where $|R_3| = |L_3|$, we refer to (16) since L and R are swapped after the reduction. For the other cases, we constrain

$$1 - w_s - 2w_r - w_c + w_b \leq 0 \quad (19) \qquad \text{and } 1 - w_s - 2w_r + w_c \leq 0. \quad (20)$$

This describes all the constraints on the measure. To minimize the running time proven by the analysis, we minimize w_r, obtaining the following optimal, feasible weights:

$$w_r = 0.2 + \varepsilon \qquad w_s = 0.7 \qquad w_{s,2} = 0.6 \qquad w_b = 0.2 \qquad w_c = 0.1.$$

All constraints are satisfied and $\mu \leq (0.2 + \varepsilon)n = (1/5 + o(1))n$.

It only remains to verify that the depth of the search trees is polynomial. Since not every reduction removes a vertex (some only modify the separation (L, S, R)), it is crucial to guarantee some kind of progress for each reduction. Since each reduction decreases another polynomially-bounded measure $\eta(L, S, R, E) := 3|S_3| + 2|R_3| + |L_3| + 2|E|$, by at least one, the depth of the search trees is indeed polynomial.

Theorem 1. *On input of a* Max 2-CSP *instance on a constraint graph G with n vertices and m edges, the described algorithm solves G in time $r^{n/5+o(n)} = r^{2m/15+o(m)}$ if G is cubic, time $r^{7m/40+o(m)}$ if G has maximum degree 4, and time $r^{9m/50+o(m)}$ in general, using polynomial space.*

This improves on the previous best running times [25] of $O^\star(r^{m/6})$, $O^\star(r^{3m/16})$, and $O^\star(r^{19m/100})$. The same improvements also hold for Max Cut, an important special case of Max $(2,2)$-CSP. Theorem 1 extends instantly to Polynomial CSP and Ring CSP, where the scores are multivariate formal polynomials, or take values in an arbitrary ring. The setting is precisely defined in [26], and the extensions follow immediately from the fact that the algorithm here depends only on R0–R3. Plugging our algorithm into the analysis of [16] also improves that running time from $O^\star(r^{n\cdot\left(1-\frac{3}{d+1}\right)})$ to $r^{n\cdot(1-\frac{3.3}{d+1})+o(n)}$ for any Max 2-CSP instance with n vertices and average degree $d \geq 5$.

3 The Separate, Measure and Conquer Technique

Our Max 2-CSP algorithm illustrates that one can exploit separator-based branching to design a more efficient exponential-time algorithm. However, Max 2-CSP algorithms have certain features that make the analysis simpler than for other problems. First, R3 produces instances with the same constraint graph, and therefore the same measure. Second, the measure of R depends only on the number of degree-3 vertices in R. This implies a discretized change in the measure for R whenever L and R are swapped. In general the change in measure when swapping L and R could take values dense within a continuous domain. Our measure for the MAX 2-CSP algorithm also implies that the initial separator only needs to balance the *number* of vertices in L and R instead of the *measure* of L and R, which is what is needed more generally. Finally, a general method is needed to combine the separator-based branching, which would typically be done for sparse instances, with the general case, where vertex degrees are arbitrary.

Our general method of analysis resolves all the complications mentioned. It applies to recursive algorithms that label vertices of a graph, and where an instance can be decomposed into two independent subinstances when all the vertices of a separator have been labeled in a certain way. Let $G = (V, E)$ be a graph and $\ell : V \to L$ be a labeling of its vertices by labels in the finite set L. For a subset of vertices $W \subseteq V$, denote by $\mu_r(W)$ and $\mu_s(W)$ two measures for the vertices in W in the graph G labeled by ℓ. The measure μ_r is used for the vertices on the right hand side of the separator and μ_s for the vertices in the separator. Let (L, S, R) be a separation of G. Initially, we use the separation $(L, S, R) = (\emptyset, \emptyset, V)$. We define the measure

$$\mu(L,S,R) = \mu_s(S) + \mu_r(R) + \max\left(0, B - \frac{\mu_r(R) - \mu_r(L)}{2}\right) + (1+B) \cdot \log_{1+\epsilon}(\mu_r(R) + \mu_s(S)), \tag{21}$$

where $\epsilon > 0$ is a constant that will be chosen small enough to satisfy constraint (24) below, and B is an arbitrary constant greater than the maximum change in imbalance in each transformation in the analysis, except the Separation transformation. The *imbalance* of an instance is $\mu_r(R) - \mu_r(L)$, and we assume, as previously, that

$$\mu_r(R) \geq \mu_r(L). \tag{22}$$

To make sure a balanced separator can be computed efficiently, we will assume that adding a vertex to R changes $\mu_r(R)$ by at most B (adjusting B if necessary):

$$|\mu_r(R \cup \{v\}) - \mu_r(R)| \leq B \qquad \text{for each } R \subseteq V \text{ and } v \in V. \tag{23}$$

We also assume that $\mu_r(R)$ can be computed in time polynomial in $|V|$ for each $R \subseteq V$.

Let us now look more closely at the measure (21). The terms $\mu_s(S)$ and $\mu_r(R)$ naturally define measures for the vertices in S and R. No term of the measure directly accounts for the vertices in L; we merely enforce that $\mu_r(R) \geq \mu_r(L)$. The term $\max\left(0, B - \frac{\mu_r(R)-\mu_r(L)}{2}\right)$ is a penalty term based on how balanced the instance is: the more balanced the instance, the larger the penalty term. The penalty term has become continuous, varying from 0 to B. The final logarithmic term amortizes the increase in measure of at most B due to the balance terms each time the instance is separated.

Let us now formulate some generic constraints that the measure should obey.

Separation. We assume that an instance with a separation (L, S, R) can be separated into two independent subinstances $(L, S, \emptyset)$ and $(\emptyset, S, R)$ when the labeling of S allows it; specifically, when all vertices in S have been labeled by a subset $L_s \subseteq L$. This arises in two cases. The first is at the beginning of the algorithm when the graph has not been separated, which is represented by the trivial separation $(\emptyset, \emptyset, V)$. The second is when our reductions have produced a separable instance.

Let (L, S, R) be such that $\ell(s) \in L_s$ for each $s \in S$. The algorithm recursively solves the subinstances $(L, S, \emptyset)$ and $(\emptyset, S, R)$. Let us focus on the instance $(\emptyset, S, R)$; the treatment of the other instance is symmetric. After a cleanup phase, where simplification rules are applied, the next step is to compute a new separator of $S \cup R$. This can be done in various ways, depending on the graph class. For example, polynomial-time computable balanced separators can be derived from upper bounds on the pathwidth of graphs with bounded maximum or average degree [6,7,12]. After a balanced separator (L', S', R') has been computed for $S \cup R$, the instance is solved recursively, and so is the instance $L \cup S$, separated into (L'', S'', R''). Both solutions are then combined into a solution for the instance $L \cup S \cup R$. Without loss of generality, assume $\mu(L', S', R') \geq \mu(L'', S'', R'')$. Assuming that the separation and combination are done in polynomial time, the imposed constraint on the measure is

$$
\begin{aligned}
&2 \cdot 2^{\mu_r(R')+\mu_s(S')+B+(1+B)\cdot\log_{1+\epsilon}(\mu_r(R')+\mu_s(S'))} \\
&\qquad \leq 2^{\mu_r(R)+\mu_s(S)+(1+B)\cdot\log_{1+\epsilon}(\mu_r(R)+\mu_s(S))} .
\end{aligned}
$$

To satisfy the constraint, it suffices to constrain that

$$
\mu_r(R) + \mu_s(S) \geq (1+\epsilon)(\mu_r(R') + \mu_s(S')). \qquad (24)
$$

This is the only constraint involving the size of a separation. It constrains that separating $(\emptyset, S, R)$ to (L', S', R') should reduce $\mu_r(R) + \mu_s(S)$ by a constant factor, namely $1 + \epsilon$.

Branching. Suppose a transformation taking (L, S, R, ℓ) to (L', S', R', ℓ') decreases $\mu_r(R) + \mu_r(L)$ by d. Since the measure includes roughly (and at least) half of $\mu_r(R) + \mu_r(L)$, ideally $\mu_r(R) + \max\left(0, B - \frac{\mu_r(R)-\mu_r(L)}{2}\right)$ decreases by $d/2$. One can show that this is indeed the case for our measure if the the following condition holds:

$$
\text{If } \mu_r(R) - \mu_r(L) > B, \text{ then } \mu_r(R) - \mu_r(R') \geq \mu_r(L) - \mu_r(L') \ . \qquad (25)
$$

Condition (25) is very natural, expressing that, if the instance is imbalanced or risks becoming imbalanced we would like to make more progress on the large side. Thus, if Condition (25) holds, then the analysis is at least as good as a non-separator based analysis, but with the additional improvement due to the separator branching.

Integration into a standard Measure and Conquer analysis. The Separate, Measure and Conquer analysis will typically be used when the instance has become sufficiently sparse that one can guarantee that a small separator exists. We can view the part played by the Separate, Measure and Conquer analysis as a subroutine with measure μ', and integrate it into any other Measure and Conquer analysis with different measure μ. We only need guarantee that the measure of an instance does not increase when transitioning to the subroutine, by constraining that $\mu'(I) \le \mu(I)$ for all instances I [12].

4 Counting Dominating Sets

The #DS problem is to compute, for a given graph G, the function d such that $d(k)$ is the number of dominating sets of G of size k. Its current fastest polynomial-space algorithm runs in time $O(1.5673^n)$ [27]. While many algorithms for domination problems rely on a transformation to Set Cover, the current fastest polynomial-space algorithm for subcubic graphs works directly on the input graph and runs in time $O^\star(2^{n/2})$ [19]. We can apply the Separate, Measure and Conquer method to design and analyze faster algorithms for #DS for subcubic graphs and, separately, for general graphs.

Theorem 2. #DS *can be solved in time* $3^{n/5+o(n)}$ *on subcubic graphs and in time* $O(1.5183^n)$ *on general graphs, using only polynomial space.*

Our algorithm for subcubic graphs uses a new 3-way branching inspired by the inclusion/exclusion branching of [28]. The algorithm of [19] had running time $O^\star(4^{n/4})$. Our 3-way branching improves its running time bound to $O^\star(3^{n/4}) = O(1.3161^n)$. Using separation improves it further to $3^{n/5+o(n)} = O(1.2458^n)$. Our algorithm for general graphs essentially just adds separation to [27].

5 Conclusions

We have presented a new method to analyze separator-based branching algorithms within the Measure and Conquer framework. It uses a novel kind of measure that amortizes the sudden large gain when an instance decomposes into independent subinstances. The key feature needed to apply the method is that an algorithm eventually reaches instances where small balanced separators can be computed efficiently. This is so for algorithms that reach sparse graphs in their final stages, but could also include cases where the treewidth of the graph is bounded, or where a graph with small treewidth can be reached by branching on a few vertices [10,15,18].

There are problems for which traditional algorithms are already so fast that branching on separators does not seem to offer an advantage. For example, the current fastest algorithm for Maximum Independent Set on subcubic graphs runs in $O(1.0836^n)$ time [29], and merely branching on the vertices of the separator would take $2^{n/6+o(n)} = \Omega(1.1225^n)$ time. A second limitation is that Separate, Measure and Conquer subroutines can often be replaced by treewidth-based dynamic programming subroutines [7], leading to the same or smaller running times; for example, #DS can be solved in time $O(1.5002^n)$ [23]. However, such algorithms use exponential space.

We believe that the Separate, Measure and Conquer method is widely applicable, but poses fresh challenges, as it presents more choices in the design of algorithms and more complications in the analysis. It also provides impetus to looking for other global properties that may be exploited to derive efficient algorithms.

Acknowledgments. The research was supported in part by the DIMACS 2006–2010 Special Focus on Discrete Random Systems, NSF grant DMS-0602942. Serge Gaspers is the recipient of an Australian Research Council Discovery Early Career Researcher Award (project number DE120101761) and a Future Fellowship (project number FT140100048). NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. The research was done in part at Dagstuhl Seminars 10441 (Exact Complexity of NP-hard problems, 2010) and 13331 (Exponential Algorithms: Algorithms and Complexity Beyond Polynomial Time, 2013), and was presented at the latter.

References

1. Achlioptas, D., Sorkin, G.B.: Optimal myopic algorithms for random 3-SAT. In: Proc. FOCS 2000, pp. 590–600 (2000)
2. Alekhnovich, M., Hirsch, E.A., Itsykson, D.: Exponential lower bounds for the running time of DPLL algorithms on satisfiable formulas. J. Autom. Reasoning **35**(1–3), 51–72 (2005)
3. Biere, A., Sinz, C.: Decomposing SAT problems into connected components. JSAT **2**(1–4), 201–208 (2006)
4. Dechter, R., Mateescu, R.: And/or search spaces for graphical models. Artif. Intell. **171**(2–3), 73–106 (2007)
5. Diestel, R.: Graph Theory. Springer (2010)
6. Edwards, K., McDermid, E.: A general reduction theorem with applications to pathwidth and the complexity of MAX 2-CSP. Algorithmica. (to appear)
7. Fomin, F.V., Gaspers, S., Saurabh, S., Stepanov, A.A.: On two techniques of combining branching and treewidth. Algorithmica **54**(2), 181–207 (2009)
8. Fomin, F.V., Grandoni, F., Kratsch, D.: A measure & conquer approach for the analysis of exact algorithms. J. ACM **56**(5) (2009)
9. Fomin, F.V.: Høie, K.: Pathwidth of cubic graphs and exact algorithms. Inform. Process. Lett. **97**(5), 191–196 (2006)
10. Fomin, F.V., Lokshtanov, D., Misra, N., Saurabh, S.: Planar F-deletion: approximation, kernelization and optimal FPT algorithms. In: Proc. FOCS 2012, pp. 470–479 (2012)
11. Freuder, E.C., Quinn, M.J.: Taking advantage of stable sets of variables in constraint satisfaction problems. In: Proc. IJCAI 1985, pp. 1076–1078 (1985)
12. Gaspers, S.: Exponential Time Algorithms - Structures, Measures, and Bounds. VDM (2010)
13. Gaspers, S., Sorkin, G.B.: A universally fastest algorithm for Max 2-Sat, Max 2-CSP, and everything in between. J. Comput. System Sci. **78**(1), 305–335 (2012)

14. Gaspers, S., Sorkin, G.B.: Separate, Measure and Conquer: Faster algorithms for Max 2-CSP and counting dominating sets (2014). arXiv:1404.0753 [cs.DS]
15. Gaspers, S., Szeider, S.: Strong backdoors to bounded treewidth SAT. In: Proc. FOCS 2013, pp. 489–498 (2013)
16. Golovnev, A., Kutzkov, K.: New exact algorithms for the 2-constraint satisfaction problem. Theor. Comput. Sci. **526**, 18–27 (2014)
17. Gottlob, G., Leone, N., Scarcello, F.: A comparison of structural CSP decomposition methods. Artif. Intell. **124**(2), 243–282 (2000)
18. Kim, E.J., Langer, A., Paul, C., Reidl, F., Rossmanith, P., Sau, I., Sikdar, S.: Linear kernels and single-exponential algorithms via protrusion decompositions. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 613–624. Springer, Heidelberg (2013)
19. Kneis, Joachim, Mölle, Daniel, Richter, Stefan, Rossmanith, Peter: Algorithms based on the treewidth of sparse graphs. In: Kratsch, Dieter (ed.) WG 2005. LNCS, vol. 3787, pp. 385–396. Springer, Heidelberg (2005)
20. Li, W., van Beek, P.: Guiding real-world SAT solving with dynamic hypergraph separator decomposition. In: Proc. ICTAI 2004, pp. 542–548 (2004)
21. Lipton, R.J., Tarjan, R.E.: Application of a planar separator theorem. In: Proc. FOCS 1977, pp. 162–170 (1977)
22. Monien, B., Preis, R.: Upper bounds on the bisection width of 3- and 4-regular graphs. J. Discrete Algorithms **4**(3), 475–498 (2006)
23. Nederlof, J., van Rooij, J.M.M., van Dijk, T.C.: Inclusion/exclusion meets measure and conquer. Algorithmica **69**(3), 685–740 (2014)
24. Scott, A.D., Sorkin, G.B.: Solving sparse random instances of Max Cut and Max 2-CSP in linear expected time. Comb. Probab. Comput. **15**(1–2), 281–315 (2006)
25. Scott, A.D., Sorkin, G.B.: Linear-programming design and analysis of fast algorithms for Max 2-CSP. Discrete Optim. **4**(3–4), 260–287 (2007)
26. Scott, A.D., Sorkin, G.B.: Polynomial constraint satisfaction problems, graph bisection, and the Ising partition function. ACM Trans. Algorithms **5**(4), Art. 45, 27 (2009)
27. van Rooij, J.M.M.: Polynomial space algorithms for counting dominating sets and the domatic number. In: Calamoneri, T., Diaz, J. (eds.) CIAC 2010. LNCS, vol. 6078, pp. 73–84. Springer, Heidelberg (2010)
28. van Rooij, J.M.M., Nederlof, J., van Dijk, T.C.: Inclusion/exclusion meets measure and conquer. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 554–565. Springer, Heidelberg (2009)
29. Xiao, M., Nagamochi, H.: Confining sets and avoiding bottleneck cases: A simple maximum independent set algorithm in degree-3 graphs. Theor. Comput. Sci. **469**, 92–104 (2013)