Fast Fusion Moves for Multi-model Estimation

Andrew Delong, Olga Veksler, and Yuri Boykov

University of Western Ontario, Canada

Abstract. We develop a fast, effective algorithm for minimizing a well-known objective function for robust multi-model estimation. Our work introduces a combinatorial step belonging to a family of powerful move-making methods like α-*expansion* and *fusion*. We also show that our subproblem can be quickly transformed into a comparatively small instance of *minimum-weighted vertex-cover*. In practice, these vertex-cover subproblems are almost always bipartite and can be solved exactly by specialized network flow algorithms. Experiments indicate that our approach achieves the robustness of methods like affinity propagation, whilst providing the speed of fast greedy heuristics.

1 Introduction

Our work is about optimizing an objective function that is of central importance to computer vision. This objective, herein denoted by $E(\cdot)$, has been used for a number of applications such as robust motion estimation [1–4], object detection [5], and geometric model fitting [6]. Intuitively, E balances the need to fit the data against the need for a 'simple' set of explanatory models. Recent work in vision has pointed out that E is also a classical *facility location* problem long studied in operations research [7–9]. So, in this work, we take it for granted that the objective is very useful, and we focus on a novel way to minimize it. However, facility location is NP-hard and, in the most general case, cannot be approximated to within any constant factor [10].

The key to our approach is a subproblem that balances computational power and tractability. Our new subproblem is an *optimized crossover* [11], or a *fusion move* [12] as more commonly known in computer vision. Though [12] proposed fusion moves for classical MAP-MRF objectives (stereo, optical flow), our fusion moves are defined on a very high-order objective; our work can be thought of as "fusion moves for facility location." To minimize E we iteratively solve our combinatorial 'fusion' subproblem, each time trying to select a better subset of models. Our subproblem considers an exponential space of solutions, so our approach is deemed a *very large-scale neighborhood search* technique [13] in the family of powerful move-making methods like α-expansion [14].

We show that our fusion operation, when applied to large data, reduces to a comparatively small *minimum-weighted vertex-cover* problem. Vertex-cover is NP-complete in general but, in model estimation scenarios, our construction is usually bipartite and can be solved by specialized *bipartite maximum-flow* algorithms [15]. Furthermore, in the non-bipartite case, there exists a fast 2-approximation algorithm for weighted vertex-cover based on elegant reduction to the easy bipartite case [16]. However, instead of relying on this reduction, we propose a new, specialized approach for the non-bipartite case with properties that are desirable within our iterative fusion framework.

The remainder of the paper is structured as follows. Section 2 defines objective E in the context of multi-model estimation, and reviews known heuristics for minimizing it.

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part I, LNCS 7572, pp. 370–384, 2012.

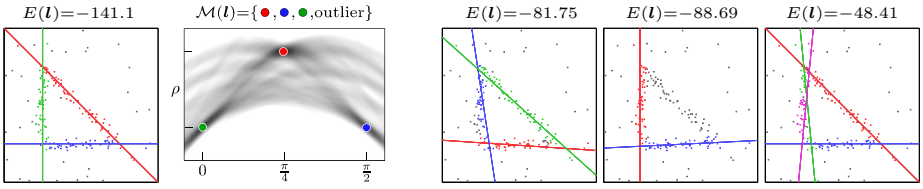


Fig. 1. An illustration of how $E(\boldsymbol{l})$ measures the quality of labeling $\boldsymbol{l}$. The low-energy labeling at left identifies the three unique lines shown in $\mathcal{M}(\boldsymbol{l})$, here superimposed on a probabilistic hough transform [18] of the data. Using model penalty $L(m) = 100$ the leftmost solution has energy $E(\boldsymbol{l}) = -441.1 + 300 = -141.1$. Three worse (higher-energy) labelings are shown at right.

Section 3 formulates our new subproblem, derives the corresponding weighted vertex-cover instance, and describes ways to solve it—including a novel construction specifically tailored to our fusion operation. Section 4 returns to multi-model estimation and describes our iterative framework: a process we call *fusion-based multi-model* RANSAC (FÜSAC) owing to the way it combines fusion with RANSAC-style random sampling [17]. Experiments indicate that our method is competitive in terms of both quality and speed. Section 5 discusses our method as it relates to greedy and genetic algorithms, and to high-order Conditional Random Fields (CRFs).

2 Multi-model Objective E

Let the observed data (*e.g.* points, correspondences) be a finite set of vectors $\{\boldsymbol{y}^i\}_{i\in\mathcal{I}}$. We seek a *labeling* $\boldsymbol{l} = (l_i)_{i\in\mathcal{I}}$ that assigns a model (*e.g.* line, plane, homography) to each individual $\boldsymbol{y}^i$. We assume each model is an element in a parameter space $\mathcal{M}$, and by assigning $l_i = m$ we indicate a belief that $\boldsymbol{y}^i$ was generated by model $m \in \mathcal{M}$. For example, if observations were generated by unbounded 2D lines, the standard (θ, ρ) line representation is a good parameter space to select from, so $\mathcal{M} = [0, 2\pi) \times \mathbb{R}_{\geq 0}$. A 'robust' version of this parameter space might incorporate a varying noise parameter $\sigma > 0$ along with an explicit *outlier* label so that $\mathcal{M} = ([0, 2\pi)\times\mathbb{R}_{\geq 0}\times\mathbb{R}_{>0})\cup\{\text{outlier}\}$.

Our objective $E(\boldsymbol{l})$ evaluates the quality of labeling $\boldsymbol{l}$ via two kinds of terms: each *likelihood* term $D_i(m)$ measures the likelihood that model m generated $\boldsymbol{y}^i$, and each *model penalty* term $L(m) \geq 0$ measures the amount of evidence needed to justify introducing model m into the solution. The main objective is

$$E(\boldsymbol{l}) \;=\; \sum_{i\in\mathcal{I}} D_i(l_i) \;+\; \sum_{m\in\mathcal{M}(\boldsymbol{l})} L(m) \tag{1}$$

where $\mathcal{M}(\boldsymbol{l}) = \{\, m \in \mathcal{M} \mid \exists l_i = m \,\}$ is the subset of models that $\boldsymbol{l}$ directly relies upon. Note that [1–6] all consider objectives of this type. If we were to set $L(m) = \lambda$, then the model penalty becomes $\lambda|\mathcal{M}(\boldsymbol{l})|$ which simply penalizes the number of unique models in a solution. In this work we always assume an outlier label $\oslash \in \mathcal{M}$ with $D_i(\oslash) = \text{const}$ and $L(\oslash) = 0$. Figure 1 shows how minima of (1) give good estimates.

For any geometric or otherwise generative model $m \in \mathcal{M}$, it is natural to define $D_i(m) = -\ln \Pr(\boldsymbol{y}^i \mid m)$. For example, in 2D line estimation the cost of assigning line $m = (\theta, \rho, \sigma)$ to a point $\boldsymbol{y}^i = (x, y)$ is naturally defined to be

$$D_i(m) = \tfrac{1}{2\sigma^2}(x\cos\theta + y\sin\theta - \rho)^2 + \ln(\sqrt{2\pi}\sigma).$$

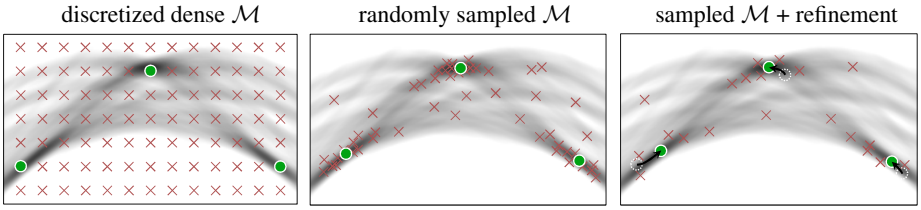


Fig. 2. Three standard approaches to finding minima of energy (1). Barinova *et al.* [5] uniformly discretize the parameter space. Pure random sampling-based methods like [2, 4] select from a sparse subset of the continuum. Refinement methods like [6] use fewer samples and iteratively descend the 'landscape' to solutions that cannot be reached by random sampling.

Our work does however allow arbitrary $D_i(\cdot)$ terms in (1). Less obvious is how to define the model penalty $L(m)$. The model penalties can be chosen by application-specific validation, or based on statistical *information critieria* such as the AIC [19] and the *geometric robust* information criterion (GRIC) [2]. The GRIC is particularly appropriate if $\mathcal{M}$ includes models of differing dimensions.

Related Work on Minimizing E. Objective (1) is like a *continuous facility location* problem but no assumptions on $D_i(\cdot)$. Hochbaum [7] gave an approximation algorithm for E when $D_i(m)$ represents a Euclidean distance from point $\boldsymbol{y}^i$ to location m, but this would constrain us to fitting point centers with isotropic noise. Variants of the general objective (1) have appeared in a number of computer vision applications, and we summarize just the most relevant work.

Torr [1] formulated two-view motion estimation as a 0-1 integer program corresponding to (1) on a finite set $\mathcal{M}$. For this application, each observation $\boldsymbol{y}^i$ is a matching pair of points, and he pre-generates candidate motions $\mathcal{M}$ by randomly sampling subsets of the matches. Minimizing (1) means selecting a small subset of rigid motions so as to minimize (for example) the reprojection error of matched points. Torr applied a generic branch-and-bound solver to compute a result. Li [4] used LP relaxation in a formulation similar to Torr's, and Schindler & Suter [3] proposed an approximation of Torr's 0-1 integer program and optimize it with the *taboo search* technique. Lazic *et al.* [20] tailored the *affinity propagation* [21] algorithm for this kind of objective.

Barinova *et al.* [5] use a special case of (1) for detecting multiple pedestrians and prominent lines in images. They work with a continuous (dense) parameter space $\mathcal{M}$ that has been uniformly discretized at sufficient resolution. They then apply the classic greedy algorithm [8] on the resulting facility location problem. Figure 2 contrasts their approach against random sampling. We mention two interesting points from their work.

1. They show that GREEDY (see Section 3), when applied in this setting, is essentially detecting peaks in a sequence of updated Hough maps (the three dark 'valleys' in Figure 1, for example). They argue that this is a better and more principled alternative to the classical *non-maximum suppression* heuristic for isolating peaks.
2. They note that, for many geometric likelihoods $D_i(\cdot)$, the dense Hough map can be incrementally updated extremely fast by specialized implementation of GREEDY.

The first point is relevant to our work because the Hough map interpretation will help to visualize and contrast our algorithm against GREEDY in Section 5. The second point

is worth noting because the generic greedy algorithm is slow if many models must be detected. By assuming that $D_i(m) < D_i(\text{outlier})$ for only a very small and quickly identifiable fraction of $i \in \mathcal{I}$, they show dramatic speed up. Their assumption holds for most scenarios (only a small fraction of points are close to each line) so it is entirely reasonable. Still, one must beware: the running time of GREEDY increases dramatically when arbitrary D_i must be assumed. We furthermore observe in Section 5 that the greedy approach has a systematic bias that limits performance on ambiguous data.

Delong *et al.* [6] briefly consider (1) for multi-model estimation. They use random sampling to generate candidates, but then they iteratively refine and re-assign the estimates in a coordinate descent scheme analogous to the k-means algorithm. As suggested in Figure 2 (right), refinement explores the solution landscape in a way that random sampling cannot. Several RANSAC variants also use refinement in one way or another, *e.g.* [22]. We incorporate refinement into our synthetic experiments as a post-processing step.

3 Our Combinatorial Subproblem (*Fusion*)

Given some fixed set $S \subseteq \mathcal{M}$, a standard way to study the facility location problem is to rewrite $E(\boldsymbol{l})$ as a *set function*

$$G(S) = \sum_{i \in \mathcal{I}} \min_{m \in S} D_i(m) + \sum_{m \in S} L(m) \tag{2}$$

where $G(\{\})$ is defined to be $+\infty$. Let $\boldsymbol{l}(S)$ denote the set of all labelings that satisfy $D_i(l_i) = \min_{m \in S} D_i(m)$ for all $i \in \mathcal{I}$. In other words, for $\boldsymbol{l} \in \boldsymbol{l}(S)$, each l_i chooses some best label from S. Now if S^* is optimal for G then any $\boldsymbol{l} \in \boldsymbol{l}(S^*)$ is optimal for E, so minimizing (2) over $S \subseteq \mathcal{M}$ is equivalent to our original objective.

The GREEDY algorithm for discrete facility location is simple. Start with an empty set S, find out which individual element $m \in \mathcal{M}$ would lead to the biggest (greediest) improvement $G(S \cup \{m\}) - G(S)$, add m to S and repeat. The pseudo-code below suggests a generic implementation, omitting efficiency-related details for simplicity.

```
GREEDY
S := {}
repeat
    for m ∈ M \ S do
        Δ[m] := G(S ∪ {m}) − G(S)
    m* := arg min_{m ∈ M\S} Δ[m]
    if Δ[m*] ≥ 0 then break
    else S := S ∪ {m*}
until S = M
return S
```

Rather than growing S greedily, we propose a completely different approach that relies on a combinatorial 'fusion' step. Section 3.1 formulates our subproblem, culminating in a reduction to *minimum-weighted vertex-cover*. Sections 3.2 and 3.3 describe standard ways—and a novel way—to attack this problem and find a good subset S. Section 4 will then use our fusion step within an iterative framework for minimizing E.

3.1 Gaining Tractability via *Fusion Constraints*

The greedy algorithm grows S monotonically, repeatedly evaluating $G(S \cup \{m\})$ to find the next element m^* to add. Ideally we should globally minimize $G(S)$ over all possible subsets $S \subseteq \mathcal{M}$, but this is NP-hard by reduction from the *set-cover* problem. We propose a compromise that remains tractable in most scenarios while minimizing over an exponential number of feasible solutions—two key criteria for powerful *very large-scale neighborhood search* (VLNS) techniques [13] such as α-expansion [14].

Given two sets $\mathcal{M}^0, \mathcal{M}^1 \subseteq \mathcal{M}$, denote a choice of 'best' labeling for each set by $\boldsymbol{l}^0 \in \boldsymbol{l}(\mathcal{M}^0)$ and $\boldsymbol{l}^1 \in \boldsymbol{l}(\mathcal{M}^1)$. The family $\mathcal{S}(\boldsymbol{l}^0, \boldsymbol{l}^1)$ that satisfies our *fusion constraints* is

$$\mathcal{S}(\boldsymbol{l}^0, \boldsymbol{l}^1) = \left\{ S \subseteq \mathcal{M}^0 \cup \mathcal{M}^1 \;\middle|\; \exists \boldsymbol{l} \in \boldsymbol{l}(S) \text{ such that } l_i \in \{l_i^0, l_i^1\} \;\forall i \in \mathcal{I} \right\}.$$

In other words, $S \in \mathcal{S}(\boldsymbol{l}^0, \boldsymbol{l}^1)$ only if a best labeling for S is a binary crossover, or *fusion*, of our choices for $\boldsymbol{l}^0$ and $\boldsymbol{l}^1$. Clearly if $S \in \mathcal{S}(\boldsymbol{l}^0, \boldsymbol{l}^1)$ then $\{l_i^0, l_i^1\} \cap S \neq \{\}$ for all $i \in \mathcal{I}$. The central subproblem that we propose to solve is

$$\text{minimize } G(S) = \sum_{i \in \mathcal{I}} \min_{m \in \{l_i^0, l_i^1\} \cap S} D_i(m) + \sum_{m \in S} L(m) \text{ over all } S \in \mathcal{S}(\boldsymbol{l}^0, \boldsymbol{l}^1). \tag{3}$$

Problem (3) can be expressed as a large high-order pseudo-boolean function and, in certain cases, can be solved via max-flow on a large graph construction similar to [6]. However, one of our main results is that (3) reduces to a vertex-cover problem of much smaller size. Attacking the vertex-cover problem can be 10–100x faster than using [6].

Theorem 1. *Given labelings $\boldsymbol{l}^0 \in \boldsymbol{l}(\mathcal{M}^0)$ and $\boldsymbol{l}^1 \in \boldsymbol{l}(\mathcal{M}^1)$, there exists an instance of minimum-weighted vertex-cover $(\mathcal{V}, \mathcal{E}, w)$ with $\mathcal{V} = \mathcal{M}^0 \cup \mathcal{M}^1$ such that $S^* \subseteq \mathcal{V}$ is an optimal cover if and only if S^* is an optimum of* (3).

Proof. We give the construction. It is useful to express (3) as a 0-1 integer program. Define $\boldsymbol{u} = \mathbf{1}_S$ to be the binary indicator function of set $S \subseteq \mathcal{M}^0 \cup \mathcal{M}^1$ so that $u_m = [m \in S]$ over $m \in \mathcal{M}^0 \cup \mathcal{M}^1$. Our problem (3) written in terms of $\boldsymbol{u}$ is simply

$$\text{minimize } G(\boldsymbol{u}) = \sum_{i \in \mathcal{I}} \min_{\substack{m \,:\, u_m = 1 \\ m \in \{l_i^0, l_i^1\}}} D_i(m) \;+ \sum_{m \,:\, u_m = 1} L(m) \tag{4}$$

$$\text{subject to } u_m + u_{m'} \geq 1 \quad \forall \{m, m'\} \in \mathcal{E} \tag{5}$$

$$\text{where } \mathcal{E} = \left\{ \{m, m'\} \mid \exists (l_i^0 = m, l_i^1 = m') \right\}.$$

Constraints (5) ensure that at least one of l_i^0 or l_i^1 must be selected, thereby retaining a one-to-one correspondence between feasible $\boldsymbol{u} = \mathbf{1}_S$ and sets $S \in \mathcal{S}(\boldsymbol{l}^0, \boldsymbol{l}^1)$. Let us use short-hand notation $D_i^0 = D_i(l_i^0)$ and $D_i^1 = D_i(l_i^1)$. Specifically, for each i there are three feasible configurations $(u_{l_i^0}, u_{l_i^1}) \in \{(1,0), (0,1), (1,1)\}$ for which the objective should pay D_i^0, D_i^1, or $\min(D_i^0, D_i^1)$ respectively. We therefore partition the sum over i into two cases and rewrite (4) as

$$= \sum_{i \,:\, D_i^0 < D_i^1} \left(D_i^1 + (D_i^0 - D_i^1) u_{l_i^0} \right) \;+ \sum_{i \,:\, D_i^0 \geq D_i^1} \left(D_i^0 + (D_i^1 - D_i^0) u_{l_i^1} \right) \;+ \sum_{m \in \mathcal{M}^0 \cup \mathcal{M}^1} L(m) u_m \tag{6}$$

Collecting coefficients in (6) until we have terms of the form $w_m \cdot u_m$ gives

$$w_m = \sum_{i \,:\, l_i^0 = m} \min(0, D_i^0 - D_i^1) \;+ \sum_{i \,:\, l_i^1 = m} \min(0, D_i^1 - D_i^0) \;+ L(m).$$

Now let $\mathcal{V} = \mathcal{M}^0 \cup \mathcal{M}^1$. After dropping the additive constant $\sum_{i\in\mathcal{I}} \max(D_i^0, D_i^1)$ from (6), our final 0-1 integer program for 'fusing' labelings $\boldsymbol{l}^0$ and $\boldsymbol{l}^1$ is

$$\begin{aligned} \text{(VC)} \quad & \text{minimize } \textstyle\sum_{m\in\mathcal{V}} w_m u_m \\ & \text{subject to } u_m + u_{m'} \geq 1 \quad \forall \{m, m'\} \in \mathcal{E} \\ & u_m \in \{0, 1\}\,. \end{aligned} \tag{7}$$

Once we remove 'trivial' variables, problem (7) is a standard instance of *minimum-weighted vertex-cover* $(\mathcal{V}, \mathcal{E}, w)$. First, if $w_m < 0$ then an optimum must assign $u_m^* = 1$, so we can remove such variables. Second, if $\mathcal{M}^0$ and $\mathcal{M}^1$ are not disjoint, then for each $m \in \mathcal{M}^0 \cap \mathcal{M}^1$ there is the possibility that some $l_i^0 = l_i^1 = m$ which causes a self-loop $(m, m) \in \mathcal{E}$. The constraint $u_m + u_m \geq 1$ forces $u_m^* = 1$ in such cases, so we remove them too. We can make the standard vertex-cover assumption that $w_m > 0$ for all $m \in \mathcal{M}^0 \cup \mathcal{M}^1$ and that $m \neq m'$ for all $\{m, m'\} \in \mathcal{E}$. □

Notice that the size of our VC problem depends only on $|\mathcal{V}|$ and $|\mathcal{E}|$, and *not* on the number of variables $|\mathcal{I}|$ needed to evaluate $G(S)$ and $E(\boldsymbol{l})$. If we assume $\mathcal{M}^0$ and $\mathcal{M}^1$ do not contain redundant labels (*i.e.* $m \in \mathcal{M}^0 \Rightarrow \exists l_i^0 = m$) then we know $|\mathcal{V}| \leq 2|\mathcal{I}|$. In any appropriate application, however, we expect $|\mathcal{V}| \ll 2|\mathcal{I}|$ since many variables will share a model. Furthermore, $\mathcal{E}$ is often sparse and bipartite (or nearly so) in practice.

In the sections that follow, we explain standard approaches, and a novel approach, for solving (7) in the context of our 'fusion' operation. We make use of the following variants of VC in our discussion:

VC-B — the tractable case when $(\mathcal{V}, \mathcal{E})$ forms a *bipartite* graph,
VC-H — where VC is relaxed to be *half-integral* with $u_m \in \{0, \frac{1}{2}, 1\}$,
VC-L — where VC is relaxed to a *linear program* with $u_m \in [0, 1]$, and
VC-A — where VC is *approximated* by an over-penalized instance of VC-B.

However, regardless of one's method of solving vertex-cover, Theorem 1 suggests a fast transformation from problem (3) into problem (7), outlined by the pseudo-code below.

FUSE($\boldsymbol{l}^0, \boldsymbol{l}^1$) where we let $\mathcal{M}^0 = \mathcal{M}(\boldsymbol{l}^0)$ and $\mathcal{M}^1 = \mathcal{M}(\boldsymbol{l}^1)$.

```
w_m := L(m)  ∀m ∈ M^0 ∪ M^1
for i ∈ I let m := l_i^0, m' := l_i^1 and do
    Δ := D_i(m') − D_i(m)
    if      Δ > 0 then  w_m  := w_m  − Δ
    else if Δ < 0 then  w_m' := w_m' + Δ
    E := E ∪ {{m, m'}}
S := MIN-WEIGHT-VC(V, E, w)  where  V := M^0 ∪ M^1
return l  where  l_i = arg min_{m∈{l_i^0, l_i^1}∩S} D_i(m)
```

3.2 Bipartite Minimum-Weighted Vertex-Cover

If graph $(\mathcal{V}, \mathcal{E})$ is bipartite, then the resulting minimum-weighted vertex-cover problem (VC-B) is tractable. VC-B can be solved by reduction to *s-t minimum cut* on an appropriately defined graph. Hochbaum [23] diagrammatically presented the reduction we describe but for the equivalent *bipartite node biclique problem*. In our case the idea is to first convert each $u_m + u_{m'} \geq 1$ constraint into a huge penalty $\eta \cdot \overline{u}_m \overline{u}_{m'}$ in the objective

function (choose $\eta = \infty$). In the bipartite case we can assume $\mathcal{M}^0$ and $\mathcal{M}^1$ disjoint (ignoring 'trivial' variables). Take one group of variables, say $\mathcal{M}^1$, and reverse their meaning with respect to $G(S)$ so that for $m \in \mathcal{M}^1$ we instead define $u_m = [m \notin S]$. The resulting unconstrained objective function corresponding to (7) is

$$G(\boldsymbol{u}) = \sum_{m\in\mathcal{M}^0} w_m u_m \; + \sum_{m\in\mathcal{M}^1} w_m \overline{u}_m \; + \sum_{\{m,m'\}\in\mathcal{E}} \eta\, \overline{u}_m u_{m'} \,. \qquad (8)$$

Since the quadratic coefficients are non-negative, pseudo-boolean function (8) can be reduced to s-t min-cut using the standard construction [24] (see [25] or §6.1 of [26]).

Once objective (8) is formulated as s-t min-cut, any number of maximum flow algorithms can solve the problem in strongly polynomial time. Since our network is bipartite (two 'layers' of nodes) many standard max-flow algorithms may run asymptotically faster [27]. There are even specialized *bipartite max-flow* algorithms [15] but, in our experiments, max-flow was not a bottleneck so we use Boykov-Kolmogorov [28].

3.3 General Minimum-Weighted Vertex-Cover

If graph $(\mathcal{V}, \mathcal{E})$ is not bipartite, we assume VC is intractable. There are two natural ways to proceed: either find an approximate solution, or find an exact solution to a problem that approximates VC. We first describe an interesting approximation algorithm based on constructing an instance of VC-B (reduction to the bipartite case). We then propose an alternative VC-B construction to approximate the VC for a fusion move.

Approximate Solution. It is easy to find a solution to VC within a factor of 2 of the optimum, but this is conjectured to be the best possible bound and reaching a factor of 1.36 is known to be NP-hard [29]. The classic approach for 2-approximation is to relax $u_m \in \{0, 1\}$ to $x_m \in [0, 1]$, solve the resulting linear program (VC-L), and simply set $u^*_m = [x^*_m \geq \frac{1}{2}]$. However, Nemhauser & Trotter [16] discovered a much more interesting approach for the equivalent problem of *maximum-weight vertex-packing*. They proved a number of striking results, many of which are summarized in [30] in the context of vertex-cover. We note that their discoveries have since been subsumed into more general *quadratic pseudo-boolean optimization* (QPBO) theory, first in [31] and recently as a standard tool for more general problems in vision [32].

Theorem 2 ([16]). *Let* VC-H *denote the* half-integral *relaxation of* VC*, where constraint $u_m \in \{0, 1\}$ is relaxed to $x_m \in \{0, \frac{1}{2}, 1\}$. Then an optimal solution to* VC-H *is also an optimal solution to full relaxation* VC-L.

Theorem 2 is especially important because Nemhauser & Trotter showed that VC-H reduces to an instance of VC-B using the construction that follows.

We are given an instance of VC-H with half-integral variables $\boldsymbol{x} = \{x_m\}_{m\in M}$. Create two sets of binary variables $\boldsymbol{z}^A = \{z^A_m\}_{m\in M}$ and $\boldsymbol{z}^B = \{z^B_m\}_{m\in M}$ for the VC-B problem. Assign the original weight w_m to both z^A_m and z^B_m. For each original edge $\{x_m, x_{m'}\} \in \mathcal{E}$ add two edges, $\{z^A_m, z^B_{m'}\}$ and $\{z^A_{m'}, z^B_m\}$, to the VC-B edge set. The following theorem establishes a one-to-one correspondence between (integral) optima of this VC-B instance and (half-integral) optima of the original VC-H instance.

Theorem 3 ([16]). *If $\boldsymbol{z}$ is feasible for* VC-B*, then $\boldsymbol{x} = \frac{1}{2}(\boldsymbol{z}^A + \boldsymbol{z}^B)$ is feasible for* VC-H. *Furthermore $\boldsymbol{x}$ is optimal for* VC-H *if and only if $\boldsymbol{z}$ is optimal for* VC-B.

To summarize: given an instance of non-bipartite VC, relax it to VC-H, reduce that to VC-B, solve with bipartite max-flow, construct a solution to VC-H according to Theorem 3, and round upwards to get a 2-approximation [30]. A real implementation could transform directly from VC to max-flow and back, but these explicit steps justify the approach. Note that this does *not* give us a 2-approximation for minimizing $G(S)$ over all (arbitrary) subsets $S \subseteq \mathcal{M}$. Again, a key advantage is that a bipartite max-flow algorithm is much, much faster than a generic linear program solver applied to VC-L.

It is worth noting that the popular QPBO method inherits its valuable *weak partial optimality* property directly from the Nemhauser-Trotter reduction.

Theorem 4 ([16]). *Suppose $\boldsymbol{x}^*$ is a (half-integral) optimum of* VC-H. *Then there exists a (binary) optimum $\boldsymbol{u}^*$ of* VC *where $u^*_m = x^*_m$ for all m with $x^*_m \in \{0, 1\}$.*

In practice this means VC-H might determine optimal assignments to many variables. In [16] they even suggest temporarily fixing each variable so that its optimal value may (if lucky) be revealed by the resulting relaxation, similar in spirit to QPBO-P [32].

Approximated (Over-penalized) Problem. Instead of applying an approximation algorithm for general VC, we propose a different strategy. We construct an instance of VC-B that approximates (*not* relaxes) our original VC problem, and solve exactly. The new VC-B problem retains the original objective value $G(\boldsymbol{u})$ for many feasible solutions, and computes a solution at least as good as each set $\mathcal{M}^0$ and $\mathcal{M}^1$ being fused.

Our construction assumes we know the original labelings $\boldsymbol{l}^0$ and $\boldsymbol{l}^1$ used to define fusion constraints $\mathcal{E}$ in (5). Without loss of generality, assume $\mathcal{M}^0$ and $\mathcal{M}^1$ contain no trivial variables. Create two sets of binary variables $\{z^0_m\}_{m\in\mathcal{M}^0}$ and $\{z^1_m\}_{m\in\mathcal{M}^1}$. Notice we only duplicate variables in $\mathcal{M}^0 \cap \mathcal{M}^1$, whereas Nemhauser-Trotter duplicates all variables. Construct the following 'over-penalized' VC-B instance:

$$\text{(VC-A)} \quad \text{minimize } \tilde{G}(\boldsymbol{z}) = \textstyle\sum_{m\in\mathcal{M}^0} w_m z^0_m \;+\; \sum_{m\in\mathcal{M}^1} w_m z^1_m \tag{9}$$

$$\text{subject to } z^0_m + z^1_{m'} \geq 1 \quad \forall (m, m') \in \tilde{\mathcal{E}} \tag{10}$$

$$\text{where } \tilde{\mathcal{E}} = \left\{ (m, m') \mid \exists (l^0_i = m, l^1_i = m') \right\}.$$

Set $\tilde{\mathcal{E}}$ is based on *ordered* pairs (m, m'), rather than the unordered pairs $\{m, m'\}$ in $\mathcal{E}$ (*i.e.* replace line **6** in FUSE). Fusion constraints (10) are indeed bipartite with respect to $\{z^0_m\}$ and $\{z^1_m\}$.

To connect $\tilde{G}(\boldsymbol{z})$ with $G(\boldsymbol{u})$ from our original VC integer program, define $\boldsymbol{u}(\boldsymbol{z})$ as

$$\boldsymbol{u}(\boldsymbol{z})_m = \begin{cases} z^0_m & \text{if } m \in \mathcal{M}^0 \setminus \mathcal{M}^1 \\ z^1_m & \text{if } m \in \mathcal{M}^1 \setminus \mathcal{M}^0 \\ z^0_m + z^1_m - z^0_m z^1_m & \text{if } m \in \mathcal{M}^0 \cap \mathcal{M}^1. \end{cases}$$

Lemma 1. *If $\boldsymbol{z}$ is feasible for* VC-A, *then $\boldsymbol{u}(\boldsymbol{z})$ is feasible for* VC.

Proof. Since $\{m, m'\} \in \mathcal{E}$ implies at least one of (m, m') or (m', m) is in $\tilde{\mathcal{E}}$, it is enough to show that $\boldsymbol{u}(\boldsymbol{z})_m + \boldsymbol{u}(\boldsymbol{z})_{m'} \geq z^0_m + z^1_{m'}$ holds for any ordered pair $(m, m') \in \tilde{\mathcal{E}}$. Since $m \in \mathcal{M}^0$ and $m' \in \mathcal{M}^1$, there are four cases to consider based on membership in $\mathcal{M}^0 \cap \mathcal{M}^1$. If $m, m' \notin \mathcal{M}^0 \cap \mathcal{M}^1$, then $\boldsymbol{u}(\boldsymbol{z})_m + \boldsymbol{u}(\boldsymbol{z})_{m'} = z^0_m + z^1_{m'}$ and the inequality holds. If $m \in \mathcal{M}^0 \cap \mathcal{M}^1$ and $m' \notin \mathcal{M}^0 \cap \mathcal{M}^1$, then $\boldsymbol{u}(\boldsymbol{z})_m + \boldsymbol{u}(\boldsymbol{z})_{m'} = z^0_m + z^1_{m'} + (z^1_m - z^0_m z^1_m) \geq z^0_m + z^1_{m'}$. The other two cases hold similarly. □

Lemma 2. *$G(\boldsymbol{u}(\boldsymbol{z})) \leq \tilde{G}(\boldsymbol{z})$ for all $\boldsymbol{z}$. Furthermore, when $\boldsymbol{z}$ satisfies $z^0_m + z^1_m \leq 1$ for all $m \in \mathcal{M}^0 \cap \mathcal{M}^1$ then $G(\boldsymbol{u}(\boldsymbol{z})) = \tilde{G}(\boldsymbol{z})$.*

Proof. Writing out $G(\boldsymbol{u}(\boldsymbol{z}))$ in terms of $\boldsymbol{z}$ gives

$$\begin{aligned} &= \sum_{m \in \mathcal{M}^0 \setminus \mathcal{M}^1} w_m z^0_m + \sum_{m \in \mathcal{M}^1 \setminus \mathcal{M}^0} w_m z^1_m + \sum_{m \in \mathcal{M}^0 \cap \mathcal{M}^1} w_m (z^0_m + z^1_m - z^0_m z^1_m) \\ &= \tilde{G}(\boldsymbol{z}) \; - \textstyle\sum_{m \in \mathcal{M}^0 \cap \mathcal{M}^1} w_m z^0_m z^1_m \,. \end{aligned} \tag{11}$$

We assume non-trivial variables with $w_m > 0$ and so $G(\boldsymbol{u}(\boldsymbol{z})) \leq \tilde{G}(\boldsymbol{z})$. If $z^0_m + z^1_m \leq 1$ for all $m \in \mathcal{M}^0 \cap \mathcal{M}^1$ then the extra terms in (11) vanish. □

Theorem 5. *If $\boldsymbol{z}^*$ is an optimal solution for* VC-A*, then $G(\boldsymbol{u}(\boldsymbol{z}^*)) \leq G(\mathcal{M}^0)$ and $G(\boldsymbol{u}(\boldsymbol{z}^*)) \leq G(\mathcal{M}^1)$, and furthermore $\boldsymbol{u}(\boldsymbol{z}^*)$ is feasible for* VC.

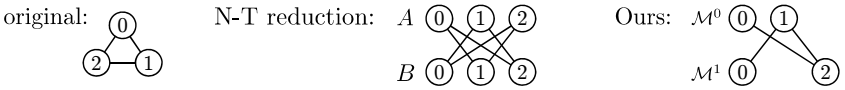
Proof. We prove the inequality for $G(\mathcal{M}^0)$. Let $\hat{\boldsymbol{z}}$ be assigned $\hat{z}^0_m = 1$, $\hat{z}^1_m = 0$ so that, by design, $\tilde{G}(\hat{\boldsymbol{z}}) = \sum_{m \in \mathcal{M}^0} w_m = G(\mathcal{M}^0)$. Also, $\hat{\boldsymbol{z}}$ is feasible with respect to bipartite fusion constraints (10), so $\tilde{G}(\boldsymbol{z}^*) \leq \tilde{G}(\hat{\boldsymbol{z}})$. By combining this with Lemma 2 we get $G(\boldsymbol{u}(\boldsymbol{z}^*)) \leq \tilde{G}(\boldsymbol{z}^*) \leq \tilde{G}(\hat{\boldsymbol{z}}) = G(\mathcal{M}^0)$. Finally, $\boldsymbol{u}(\boldsymbol{z}^*)$ is feasible for VC by Lemma 1. The case for $G(\mathcal{M}^1)$ holds by symmetric argument. □

Since VC-A is bipartite (an instance of VC-B) we can express it as s-t minimum cut and solve using a fast bipartite maximum flow algorithm, as mentioned in Section 3.2.

Theorem 5 implies a monotonicity property similar to applying QPBO-I [32]. To motivate our over-penalized construction over classical VC approximation algorithms like N-T, consider the following.

Observation 1. *Suppose we construct an instance of* VC *for fusing sets $\mathcal{M}^0$ and $\mathcal{M}^1$. Let $\boldsymbol{x}^*$ be a (half-integral) solution to* VC-H*, and let $\lceil \boldsymbol{x}^* \rceil$ be a binary solution by upward rounding. It is possible that $G(\lceil \boldsymbol{x}^* \rceil) > G(\mathcal{M}^0)$ and/or $G(\lceil \boldsymbol{x}^* \rceil) > G(\mathcal{M}^1)$, i.e. the computed solution may be worse than either of the original sets being fused.*

We show by example. Imagine we wish to fuse labelings $\boldsymbol{l}^0 = (0, 1, 1)$ and $\boldsymbol{l}^1 = (2, 2, 0)$. Then $\mathcal{M}^0 = \{0, 1\}$, $\mathcal{M}^1 = \{0, 2\}$ and $\mathcal{E} = \{\{0, 1\}, \{0, 2\}, \{1, 2\}\}$, and let us also suppose the VC weights are all $w(\cdot) = 1$. Below we see the structure of the original VC problem, the Nemhauser-Trotter (QPBO) reduction, and our approximation.



By inspection, a min-weighted vertex-cover for the N-T reduction must select either row A or row B and so $\boldsymbol{x}^* = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. However, this gives $G(\lceil \boldsymbol{x}^* \rceil) = 3$ and since both $G(\mathcal{M}^0) = G(\mathcal{M}^1) = 2$ we have a case where relaxation gives a worse result.

Besides the convenient monotonicity implied by Theorem 5, note that VC-A tends to have much fewer variables and constraints than N-T reduction, especially when $|\mathcal{M}^0 \cap \mathcal{M}^1| \ll |\mathcal{M}^0 \cup \mathcal{M}^1|$.

4 Minimizing E with Fusion and Random Sampling

Now that we have algorithmic tools to implement the FUSE operation in Section 3.1, we return to the main application under consideration: robust multi-model estimation. We aim for a method that, simultaneously,

1. is progressive (*i.e.* a 'current best solution' is always available, like RANSAC),
2. explicitly minimizes a real objective function, in this case E , like [1–6, 20],
3. achieves the speed of heuristic approaches, like greedy [5], yet
4. has the combinatorial power of full-blown VLNS [13] methods like [14, 12].

We propose combining fusion with random sampling in a process we call *fusion-based multi-model* RANSAC (FÜSAC). The basic version of our algorithm is progressive in that each candidate model $m \in \mathcal{M}$ is *immediately* accepted or rejected based on the result of a fusion attempt. This 'online' approach stands in contrast to multi-model methods [1–6] where a large set of candidates (elements of $\mathcal{M}$) is first precomputed and then either pruned greedily or used to build a monolithic 0-1 integer program.

FÜSAC(t_{max}) where t_{max} is time limit

1 $\boldsymbol{l}^0 := \oslash$ (start with just outlier label)
2 **repeat**
3 $m_{\text{new}} :=$ parameters determined by *minimal sample subset* selected from $\{\boldsymbol{y}^i\}_{i \in \mathcal{I}}$
4 $\boldsymbol{l}^1 := \boldsymbol{l}(\{\oslash, m_{\text{new}}\})$ (choose a best labeling with $\mathrm{outlier}$ and m_{new})
5 $\boldsymbol{l}^0 :=$ FUSE($\boldsymbol{l}^0, \boldsymbol{l}^1$)
6 **until** time exceeds t_{max}
7 **return** $\boldsymbol{l}^0$

This "fuse one model at a time in arbitrary order" strategy is often close to the greedy algorithm in terms of effectiveness, but poor local minima are still possible. If poor candidates are selected in the early stages, simply because they were the best 'so far', a better candidate may not be enough to correct these mistakes when it is presented. One way to fix such mistakes is by fusing two or more good candidates *simultaneously*. So, to take better advantage of the combinatorial power of our fusion step, we want labelings $\boldsymbol{l}^0$ and $\boldsymbol{l}^1$ to both contain many models so that FUSE may 'stitch' the best configurations together. We suggest a simple *genetic* variant (FÜSAC-GA) that generates multiple independent solutions and performs fusion, or *optimized crossover* [11], to find better subsets of models from among that population.

FÜSAC-GA($n_{\text{pop}}, t_{\text{max}}$) where n_{pop} is population size, t_{max} is time limit for each

1 $\boldsymbol{l}^0 :=$ FÜSAC(t_{max})
2 **for** $2..n_{\text{pop}}$ **do**
3 $\boldsymbol{l}^1 :=$ FÜSAC(t_{max}) (for efficiency, just recycle permuted models sampled in line **1**)
4 $\boldsymbol{l}^0 :=$ FUSE($\boldsymbol{l}^0, \boldsymbol{l}^1$) (fuse / optimized crossover)
5 **return** $\boldsymbol{l}^0$

Bear in mind that the FUSE operation can in principle be applied to any two candidate labelings, regardless of how they were generated. For example, [12] runs a diverse set of heuristic optical flow methods, then fuses them together. All this is parallelizable.

Figure 3 shows that the greedy strategy is consistently prone to certain mistakes, even if it can choose the best model from the entire dense parameter space. In other words, on hard problems, GREEDY may choose a particular *worse* solution as we allow more candidates. So, irrespective of speed, GREEDY may simply fail to minimize E. It is

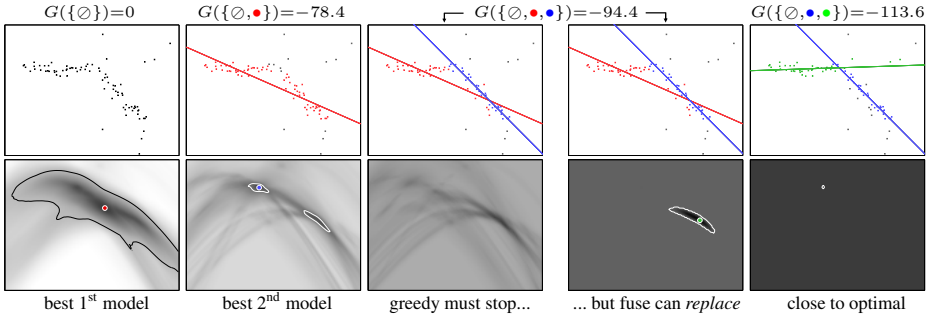


Fig. 3. Example of how fusion is more powerful than monotonic greedy. First row shows current solution $S \subset \mathcal{M}$, and below (columns 1–3) shows value $G(S \cup \{m\})$ for all $m = (\theta, \rho)$. A contour identifies region $G(S \cup \{m\}) < G(S)$, and the highlighted point shows a greedy choice for m. Columns 4–5 show value $G(\textsc{Fuse}(\boldsymbol{l}^0, \boldsymbol{l}^1))$ where $\boldsymbol{l}^0 \in \boldsymbol{l}(S)$ and $\boldsymbol{l}^1 \in \boldsymbol{l}(\{\oslash, m\})$ for all $m = (\theta, \rho)$. In contrast to column 3, fusion considers the opportunity to *replace* subsets of lines.

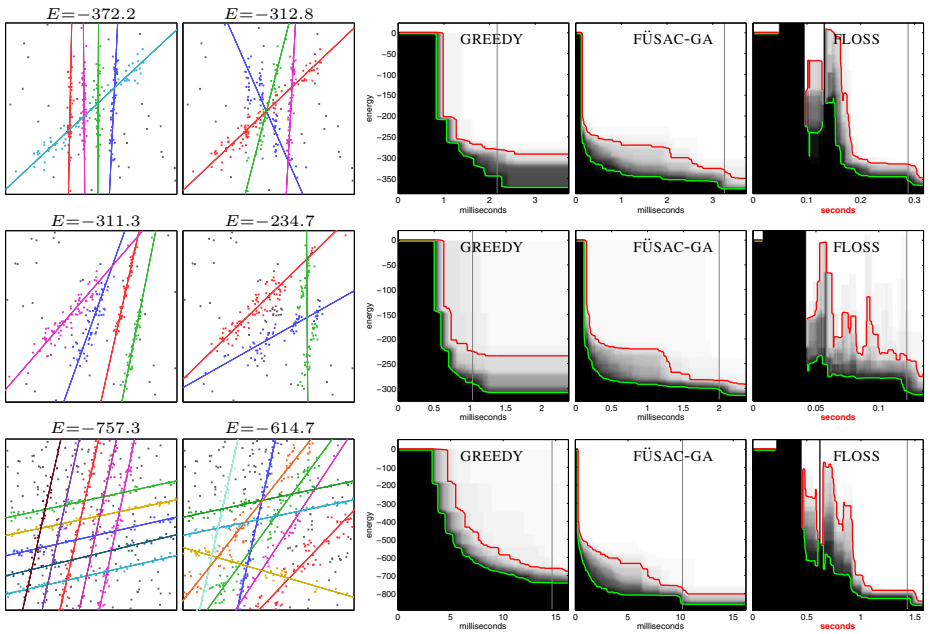


Fig. 4. Synthetic examples (one per row). Each shows a typical low-energy and high-energy solution. Performance is shown by a probabilistic time-vs-energy plot averaged over 50 trial runs. Top curve (red) indicates worst performance (E is below with 90% confidence), and the bottom curve (green) indicates best performance (E is below with only 10% confidence). The vertical gray line marks time of completion, when refinement begins. Notice on 3rd row that, even when GREEDY has many enough candidates, it may systematically choose sub-optimal solutions, repeatedly.

long known that applying optimized crossover among 'diverse' solutions is an effective attack on NP-hard problems [11, 12]. FÜSAC-GA is the first attempt to use this strategy on E (facility location), taking advantage of the fact that FÜSAC is non-deterministic.

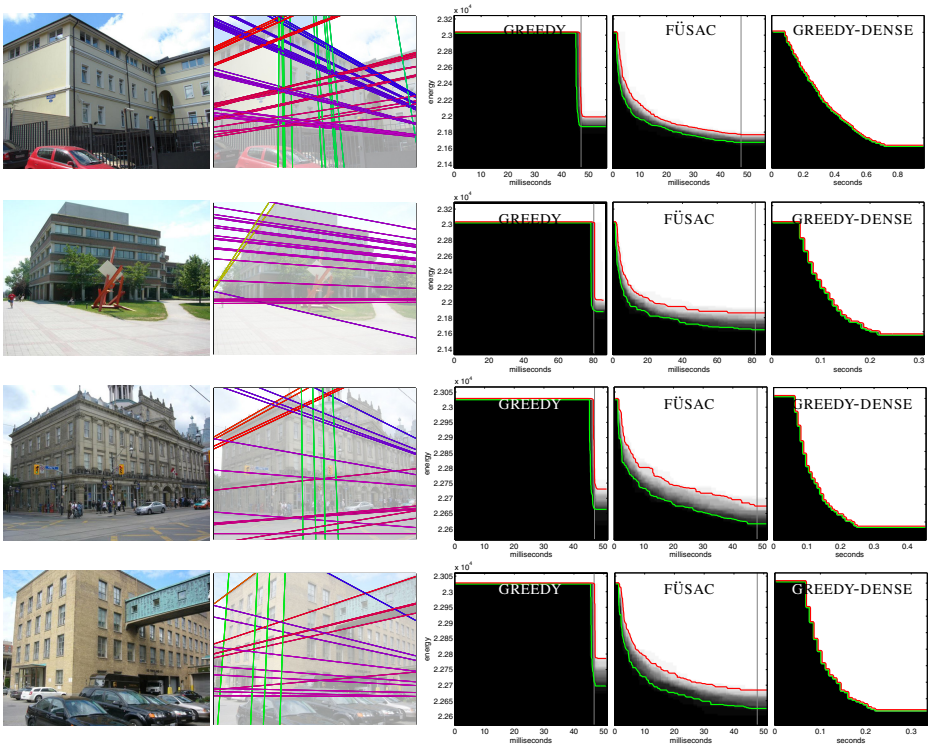


Fig. 5. Detecting edge structures in real images. Each row is an example with input, typical output (at left), and a probabilistic time-vs-energy plot for each method. GREEDY-DENSE [5] is most reliable at reaching a low energy, but sampling-based methods (GREEDY, FÜSAC) are significantly faster. GREEDY spends the majority of its time first computing $D_i(\cdot)$ for all candidates; all methods were given the same set of candidates, so this implies inference is not the bottleneck for GREEDY nor FÜSAC. Computing $D_i(\cdot)$ could be done on a GPU. Given more time the sampling-based methods do not improve significantly, likely because smarter sampling is needed (*e.g.* sample two collinear edges). FLOSS (not shown) converges to solutions similar to FÜSAC in 1–3 seconds; FLOSS is quite robust, so this is further evidence that sampling is the limiting factor.

We now show some of our experimental results, focusing on comparison to the greedy approach. Barinova *et al.* [5] found that the greedy approach outperforms several popular alternatives such as non-maximum suppression, medoid-shift [33], and LP-relaxation [4]. We also include results for our C++ implementation[1] of *facility location affinity propagation* (FLoSS) [20]. C++/MATLAB code for our own algorithm will be made available at http://vision.csd.uwo.ca/code/.

There are also many optimizations for RANSAC to consider, but most do not directly extend to a multi-model setting (*e.g.* 'bail out' schemes). We use 'blind' sampling, but the performance of our framework depends directly on how much time is wasted evaluating bad samples; guided sampling [34] would likely improve our results, for example the recent work of Pham *et al.* [35].

[1] Messages were computed efficiently; 50 iterations per test, with damping factor $\lambda = 0.75$; all decoding times were excluded, so time plots reflect time for message updates only.

Experiments on Synthetic Data. First, we aim to show that FÜSAC is competitive with a generic implementation of GREEDY in terms of quality and speed. The FLOSS algorithm is highly robust, but takes a long time to converge. Second, we show that FÜSAC-GA (*genetic algorithm* variant) is more robust at escaping local minima in challenging instances. We apply these techniques to line estimation. Each candidate model $m = (\theta, \rho, \sigma)$ is generated by fitting θ and ρ to two randomly sampled points, and choosing random $\sigma \in [0.01, 0.05]$. Figure 4 shows typical results on non-trivial instances, though on unambiguous instances all methods succeed and are comparable.

Experiments on Real Images. We validate the basic idea of FÜSAC by showing that it is competitive at detecting edge structures in images from the York Urban dataset [36]. The idea is to detect oriented edge features $\boldsymbol{y}^i = \{x, y, \alpha\}$ based on local gradients, and find clusters of features that are collinear and consistently oriented. Such detections are useful, for example, within a larger geometric image parsing framework. Barinova *et al.* already showed that, for detecting edge structures, minima of objective E give much better precision-recall curves than non-maximum suppression and medoid-shift. Our paper is mainly about optimization, so it is sufficient to show that our method is competitive in that respect. We used their second code release (GREEDY-DENSE) and followed their default set up with $D_i(m) = \text{coeff1} \cdot |x \cos\theta + y \sin\theta - \rho| + \text{coeff2} \cdot |\alpha - \theta|$. For each image, we randomly select up to 5,000 edge features to be used in detection. Each candidate model is generated by randomly sampling one edge feature, and using its orientation α to define the line. Like [5] our code takes advantage of the fact that $D_i(m) < D_i(\text{outlier})$ for only a sparse subset of points[2]. Figure 5 shows a few results. Performance is similar to theirs across the dataset, suggesting these examples are 'easy' for all algorithms considered (FÜSAC-GA performs similarly to FÜSAC).

5 Conclusion

We proposed a fast combinatorial approach for minimizing a central objective function in vision. Our strategy is much like the powerful move-making methods [14, 12] and, to the best of our knowledge, we are the first to formulate a fusion operation for this important objective. Furthermore, we showed how to quickly transform fusion problems of size $O(\mathcal{I})$ into vertex-cover problems of size $O(\mathcal{M}^0 \cup \mathcal{M}^1)$, and proposed an alternate construction for the (rare) non-bipartite case. Our algorithm is quite fast, yet seem to be competitive with state-of-the-art methods like affinity propagation.

References

1. Torr, P.H.S., Murray, D.: Stochastic Motion Clustering. In: Eklundh, J.-O. (ed.) ECCV 1994, Part II. LNCS, vol. 801, pp. 328–337. Springer, Heidelberg (1994)
2. Torr, P.H.S.: Geometric Motion Segmentation and Model Selection. In: Philosophical Transactions of the Royal Society A, pp. 1321–1340 (1998)
3. Schindler, K., Suter, D.: Two-view multibody structure-and-motion with outliers through model selection. IEEE Trans. on Patt. Analysis and Mach. Intelligence 28, 983–995 (2006)

[2] Maintaining sparse list of inliers can speed up by roughly 2–6x, and was *not* applied to FLOSS.

4. Li, H.: Two-view Motion Segmentation from Linear Programming Relaxation. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2007)
5. Barinova, O., Lempitsky, V., Kohli, P.: On the Detection of Multiple Object Instances using Hough Transforms. In: IEEE Conf. on Comp. Vision and Patt. Recognition, CVPR (2010)
6. Delong, A., Osokin, A., Isack, H.N., Boykov, Y.: Fast Approximate Energy Minimization with Label Costs. International Journal of Computer Vision (IJCV) 96, 1–27 (2011)
7. Hochbaum, D.S.: Heuristics for the fixed cost median problem. Math. Prog. 22 (1982)
8. Cornuejols, G., Nemhauser, G.L., Wolsey, L.A.: The Uncapacitated Facility Location Problem. Technical Report 605, Operations Research, Cornell University (1983)
9. Shmoys, D.B., Tardos, E., Aardal, K.: Approximation algorithms for facility location problems. In: ACM Symposium on Theory of Computing (STOC), pp. 265–274 (1998)
10. Feige, U.: A Threshold of $\ln n$ for Approximating Set Cover. Jour. of the ACM 45 (1998)
11. Aggarwal, C.C., Orlin, J.B., Tai, R.P.: Optimized Crossover for the Independent Set Problem. Operations Research 45, 226–234 (1997)
12. Lempitsky, V., Rother, C., Roth, S., Blake, A.: Fusion moves for markov random field optimization. IEEE Transactions on Pattern Analysis and Machine Inference (TPAMI) 32 (2010)
13. Ahuja, R.K., Ergun, O., Orlin, J.B., Punnen, A.P.: A survey of very large-scale neighborhood search techniques. Discrete Applied Mathematics 123, 75–202 (2002)
14. Boykov, Y., Veksler, O., Zabih, R.: Fast Approximate Energy Minimization via Graph Cuts. IEEE Transactions on Pattern Recognition and Machine Intelligence (TPAMI) 23 (2001)
15. Ahuja, R., Orlin, J., Stein, C., Tarjan, R.: Improved algorithms for bipartite network flow. SIAM Journal on Computing 23, 906–933 (1994)
16. Nemhauser, G., Trotter, L.: Vertex packings: Structural properties and algorithms. Mathematical Programming 8, 232–248 (1975)
17. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Comm. ACM 24, 381–395 (1981)
18. Stephens, R.: Probabilistic approach to the Hough transform. Image and Vis. Comp. 9 (1991)
19. Akaike, H.: A new look at statistical model identification. Trans. on Auto. Control 19 (1974)
20. Lazic, N., Givoni, I., Frey, B.J., Aarabi, P.: FLoSS: Facility Location for Subspace Segmentation. In: International Conference on Computer Vision, ICCV (2009)
21. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science 315, 972–976 (2007)
22. Chum, O., Matas, J., Kittler, J.: Locally Optimized RANSAC. Pattern Recognition (2003)
23. Hochbaum, D.: Approximating clique and biclique problems. Jour. of Algorithms 29 (1998)
24. Hammer, P.L.: Some network flow problems solved with pseudo-boolean programming. Operations Research 13, 388–399 (1965)
25. Kolmogorov, V., Zabih, R.: What Energy Functions Can Be Optimized via Graph Cuts. IEEE Transactions on Pattern Recognition and Machine Intelligence (TPAMI) 26, 147–159 (2004)
26. Boros, E., Hammer, P.L.: Pseudo-Boolean Optimization. Discrete Applied Math. 123 (2002)
27. Gusfield, D., Martel, C., Fernandez-Baca, D.: Fast algorithms for bipartite network flow. SIAM Journal on Computing 16, 237–251 (1987)
28. Boykov, Y., Kolmogorov, V.: An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. IEEE Transactions on Pattern Recognition and Machine Intelligence (TPAMI) 29, 1124–1137 (2004)
29. Dinur, I., Safra, S.: The importance of being biased. In: ACM STOC (2002)
30. Chlebík, M., Chlebíková, J.: Crown reductions for the Minimum Weighted Vertex Cover problem. Discrete Applied Mathematics 156, 292–312 (2008)

31. Hammer, P., Hansen, P., Simeone, B.: Roof duality, complementation and persistency in quadratic 0-1 optimization. Mathematical Programming 28, 121–125 (1984)
32. Rother, C., Kolmogorov, V., Lempitsky, V., Szummer, M.: Optimizing Binary MRFs via Extended Roof Duality. In: IEEE Conf. on Comp. Vis. and Patt. Recognition, CVPR (2007)
33. Sheikh, Y., Khan, E., Kanade, T.: Mode-seeking by medoidshifts. In: ICCV (2007)
34. Tordoff, B., Murray, D.W.: Guided Sampling and Consensus for Motion Estimation. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part I. LNCS, vol. 2350, pp. 82–96. Springer, Heidelberg (2002)
35. Pham, T.T., Chin, T.J., Yu, J., Suter, D.: The Random Cluster Model for Robust Geometric Fitting. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2012)
36. Denis, P., Elder, J.H., Estrada, F.J.: Efficient Edge-Based Methods for Estimating Manhattan Frames in Urban Imagery. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 197–210. Springer, Heidelberg (2008)