DEAD-END ELIMINATION AS A HEURISTIC FOR MIN-CUT IMAGE SEGMENTATION

Mala L. Radhakrishnan[*†], *Sara L. Su*[†]

MIT [*]Department of Chemistry and [†]Computer Science and Artificial Intelligence Laboratory

ABSTRACT

We apply the dead-end elimination (DEE) strategy from protein design as a heuristic for the max-flow/min-cut formulation of the image segmentation problem. DEE combines aspects of constraint propagation and branch-and-bound to eliminate solutions incompatible with global optimization of the objective function. Though DEE can be used for segmentation into an arbitrary number of regions, in this paper we evaluate only the case of binary segmentation. We provide a runtime analysis and evaluation of DEE applied to two min-cut algorithms. Preliminary results show that DEE consistently reduces the search space for the Edmonds–Karp algorithm; tuning DEE as a heuristic for Boykov–Kolmogorov and other algorithms is future work.

Index Terms— Image segmentation, graph theory

1. INTRODUCTION

A number of successful graph-cut techniques have been proposed for image segmentation, including the methods of Wu and Leahy [1], Shi and Malik [2], and Boykov, Jolly, and Kolmogorov [3, 4, 5]. These methods model an image as a graph of pixels and their pairwise similarities. In the graph $G = (V, E)$, nodes V correspond to pixels, and the capacity of edge $(i, j) \in E$ is proportional to the similarity of the pixels represented by nodes i and j. The graph's terminals s and t represent the foreground and background regions. Each node is connected to s and t with edges whose weights measure the pixel's "foregroundness" and "backgroundness".

It has been shown that the *s–t minimum cut* of such a graph yields an optimal binary partitioning based on a pairwise objective function [3]. The goal is to cut edges to form two disjoint sets such that similar nodes are in the same set and dissimilar nodes are in different sets. The cost of the cut is the sum of the capacities of the cut edges, and the minimum s-t cut of this graph minimizes the energy function

$$E = \sum_{i \in V} self(i_p) + \sum_{i,j \in V,\ i<j} pair(i_p, j_q)\ , \qquad (1)$$

where $self(i_p)$ is the *self energy* of pixel i in state p, and p = foreground or background. $pair(i_p, j_q)$ is the *pairwise energy* between pixels i and j. When $p \neq q$ (*i.e.*, when the pixels are assigned to different segments), $pair(i_p, j_q) = w(i, j)$. If pixels are assigned to the same segment, their pairwise energy is 0. The self energies are such that the energy associated

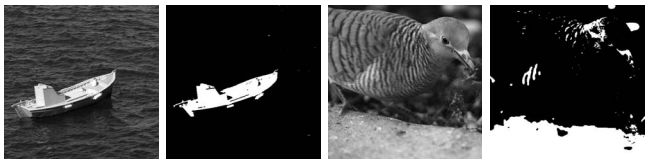


Figure 1: Segmentations of two 500×500 images using the Eq. 2 energy function. Both min-cut algorithms, alone and preceded by DEE, produce identical segmentations. DEE is able to greatly reduce the search space, regardless of the quality of the final segmentation.

with a pixel's "foregroundness" is the weight of the edge connecting its node to the background terminal, and vice versa; with these assignments, the *s-t* min-cut minimizes the above energy function. For a more detailed explanation of self and pair energies, see the paper by Boykov and Jolly [3].

In the following section, we discuss *dead-end elimination* (DEE) and its application to this minimization problem. In §3 and §4, we evaluate DEE as a preconditioner to two min-cut algorithms. Finally, we summarize our findings in §5 and consider directions for future work.

2. DEE AS A MIN-CUT PRECONDITIONER

DEE is used in computational chemistry for the combinatorial optimization problem of assigning amino acids at protein positions such that the energy of a desired protein structure is minimized. We summarize the form of the DEE theorem introduced by Goldstein [6]. (Please see the papers by Goldstein or Desmet *et al.* [7] for a more detailed explanation.) Consider a linear objective function of the form

$$E = \sum_i E(i_a) + \sum_{i<j} E(i_a, j_b)\ ,$$

where $E(i_a)$ is the self energy of assignment a in the i^{th} position and $E(i_a, j_b)$ is the pairwise energy between assignments a and b in the i^{th} and j^{th} positions, respectively.

We will optimize this function over the discrete combinatorial space of assignments at each position. This general problem is NP-hard [8]. However, if there are at most two assignments per position and the energy function is of the form outlined in §1, the problem reduces to the s-t min-cut problem of Equation 1. Let i_a and i_b be two specific assignments at position i. Then, if

$$E(i_a) - E(i_b) + \sum_j \min_f [E(i_a, j_f) - E(i_b, j_f)] > 0\ ,$$

sparse DEE singles (one iteration):
for each pixel i
  for each possible assignment i_a
    for each possible assignment $i_b \neq i_a$
      for each neighbor j of i
        choose possible assignments f to minimize $E(i_a, j_f) - E(i_b, j_f)$
      if total energy of these assignments with i_a is greater than the
      total energy of these assignments with i_b, eliminate i_a.

sparse DEE pairs (one iteration):
for each pixel i
  for each possible assignment i_a
    for each neighbor j of i
      for each possible assignment j_b
        for each possible pair of assignments i_c and j_d s.t.$((c \neq a)$ or $(d \neq b))$
          for each neighbor k of i or j
            choose possible assignments k_f to minimize the difference between
            the state energies of $(i_a + j_b + k_f)$ and $(i_c + j_d + k_f)$.
            if total state energy of these assignments with (i_a, j_b) is greater than
            total state energy with (i_c, j_d), eliminate pair (i_a, j_b).

Figure 2: DEE singles and pairs routines. "Possible" assignments (or pairs of assignments) are those that have not yet been eliminated. Note that for binary image segmentation, there are only two possible assignments per position, so the for–loops shown simplify greatly; nevertheless, DEE can be applied to problems involving any number of assignments per position.

the assignment i_a cannot be in the global minimum configuration and can therefore be eliminated from the search space. The condition asserts that i_a cannot be in the *global minimum energy assignment* (GMEA) if there exists another assignment at the same position i_b such that the total energy with i_a is higher than the total energy with i_b, even when we choose every other position to give i_a the best pairwise energies relative to i_b. Note that, in the general case, when there is an arbitrary number of possible assignments per pixel, this does not guarantee that i_b is in the GMEA, only that i_a is not. For more details and the proof of the DEE theorem, please see the supplemental technical report, available at `http://csail.mit.edu/~sarasu/pub/icip06`.

In practice, *DEE singles* and *pairs* eliminations[1] are run iteratively until either a global minimum is found or the remaining space is small enough that it can be enumerated or passed to another algorithm. In cases where few assignments can be eliminated, the solution time is worst-case exponential, a consequence of the problem being NP-hard in general protein design applications; however, DEE's eliminating power makes it extremely powerful in practice.

2.1. Application of DEE to image analysis

The min-cut formulation of the segmentation problem is identical to minimizing a pairwise objective function that utilizes the self and pair energies from §1. To cast this as a problem suitable for DEE, let the number of positions be the number of pixels. We apply DEE to eliminate assignments for pixels that are incompatible with the global energy minimum. The DEE routine (Figure 2) allows for an arbitrary number m of possible assignments per pixel. When there are only two possible states (foreground or background), eliminating an assignment for a pixel trivially assigns it.

We now analyze the runtime of DEE on images and argue that it can be used as a heuristic. Assuming that each pixel has a constant number of neighbors allows for a sparse implementation. In the sparse DEE singles routine, we loop over pixels only once. Therefore, each iteration of DEE singles takes $O(n)$ time, where n is the number of pixels, and the time to determine if a pixel can be eliminated is proportional to the number of neighbors J. A sparse implementation of DEE pairs also runs in $O(n)$ time because the number of pairs is a multiple of the number of pixels. However, now, the constant is proportional to J^2. The runtime also depends on m. From the pseudocode, we see that DEE singles and pairs run in $O(m^3)$ and $O(m^5)$ time, respectively; these factors are small for the binary case, $m = 2$.

Running a constant number of linear-time DEE iterations may greatly reduce the size of the problem, and we can feed the reduced problem into a polynomial-time algorithm such as Edmonds–Karp [9]. DEE will not increase the asymptotic running time; even if DEE eliminates nothing, the $O(n)$ running time will not asymptotically affect the polynomial running time of the min-cut, assuming a sparse graph with n nodes and $O(n)$ edges. We can alternatively run DEE singles iterations until no more pixels are assigned in two consecutive iterations. Now, the worst-case running time is $O(n^2)$, although this may yield the optimal answer faster by eliminating more of the space before the min-cut algorithm is called.

We have found that our implementation of DEE pairs provides value as a preconditioner only for certain images, as the time to run DEE pairs often outweighs its eliminating power. DEE pairs helps only because DEE singles now considers fewer pairs, so terms not eliminated before might now be eliminated. Here we focus our discussion on singles. Improving our pairs implementation is future work; see the supplementary material for further discussion.

We propose the following heuristic for min-cut image segmentation: Run DEE singles through a constant number of iterations or until no more pixels can be assigned. Output all determined pixels. For all pixels that have not yet been assigned, modify their assignments' self energies such that now,

$$E(i_a, updated) = E(i_a) + \sum_{j=D_i} E(i_a, j_d) \, ,$$

where j_d is the determined assignment of pixel j and D_i is the set of neighbors of i whose assignment has been determined. We only need to consider pair energies between two

[1] The theorem can be extended to eliminate pairs of assignments; a pair can never occur together in the global energy minimum assignment if there is another pair at those positions that is always better. The proof for this is analogous to the one for singles, where each i_a or i_b becomes a pair of assignments.

pixels that are both undetermined. A smaller graph can be constructed with these modified self energies and remaining pair energies, and a standard polynomial-time algorithm is then used on this reduced problem. The time to create the modified graph is the time to loop through all neighbors of all unassigned pixels to generate the new self energies: $O(n)$.

2.2. A problem-specific graph interpretation of DEE

DEE in protein design is generally applied to problems for which self and pair energies can take arbitrary values. In our application to image segmentation, the pair energy between two nodes (pixels) in the same state is zero, and that between pixels in different states is non-negative. This reduced energy landscape simplifies our implementation.

With these energy constraints, the DEE singles routine reduces to a simpler heuristic: For each node i on the graph corresponding to an unassigned pixel, if the sum of the edges directly leaving i that eventually lead to terminal s through at most one neighbor is greater than the sum of all edges that lead to terminal t plus the sum of all the undetermined edges (edges that can still go to both terminals), then assign i to s. The graph interpretation of DEE pairs is more complex, but again, speed-ups in the implementation can be made by making use of the energy constraints.

3. IMPLEMENTATION

We compared segmentations of grayscale natural images using two common min–cut algorithms: Edmonds–Karp (EK) [9] and Boykov–Kolmogorov [5]. We applied Goldstein DEE as a preconditioner to these and compared the running times with and without DEE. We implemented EK and DEE in C++ and used the existing fastest-known C++ implementation of BK. (All were compiled with the '-O2' flag.) Pre– and post–processing (computing edge weights and displaying the final segmented images) was done in MATLAB. We used two simple weight functions. The first is linear in the difference between the intensities I_u and I_v of two pixels u and v:

$$w(u,v) = \max(0, \lfloor c(k - |I_u - I_v|) \rfloor) \,, \tag{2}$$

where c is a real constant, $0 < c \leq 1$, and k is an integral constant, $0 < k < 255$ (*i.e.*, the range of valid pixel values). For our experiments, we chose $c = 0.25$ and $k = 100$. We also tried the exponential function

$$w(u,v) = \lfloor c_1 e^{-(I_u - I_v)^2 / c_2} \rfloor \tag{3}$$

where c_1 and c_2 are positive integral constants. Here, we set $c_1 = 25$ and $c_2 = 100$. The self energies for the two states are I_a and $(255 - I_a)$.

Note that the goal of our work was not to find the best weight function or energy function but to test a heuristic that would improve performance in practice.

Input	EK	DEE+EK	%
$bird1_{50}$	0.36	0.05	80%
$bird2_{50}$	0.26	0.02	98%
cat_{50}	0.43	0.07	80%
$coast_{50}$	0.35	0.03	92%
$feather_{50}$	0.43	0.05	90%
$fungus_{50}$	0.31	0.02	97%
$monkey_{50}$	0.31	0.01	98%
$rock1_{50}$	0.34	0.02	97%
$rock2_{50}$	0.36	0.02	95%
$rowboat_{50}$	0.31	0.02	100%
$squirrel_{50}$	0.30	0.02	95%
$windmill_{50}$	0.42	0.02	95%

Input	BK	DEE+BK	%
$bird1_{500}$	0.04	0.10	64%
$bird2_{500}$	0.02	0.09	61%
cat_{500}	0.12	0.18	57%
$coast_{500}$	0.04	0.07	74%
$feather_{500}$	0.05	0.11	64%
$fungus_{500}$	0.02	0.05	88%
$monkey_{500}$	0.03	0.04	92%
$rock1_{500}$	0.02	0.05	85%
$rock2_{500}$	0.04	0.09	76%
$rowboat_{500}$	0.02	0.03	97%
$squirrel_{500}$	0.04	0.11	51%
$windmill_{500}$	0.04	0.07	87%

Table 1: CPU times for preliminary trials with natural images (subscript 50 indicates a 50×50 image, 500 a 500×500 image). EK and BK are the times for Edmonds–Karp and Boykov–Kolmogorov alone; DEE+EK and DEE+BK are the times with DEE as a preconditioner. % is the percentage of pixels that DEE was able to assign. The energy function used is Eq. 2; Eq. 3 produced similar results. Times do not include time to compute and read in initial graph weights from a file (which, for larger images, was the dominant term but not the focus of this study) and, for BK and DEE+BK, the time to output the final segmentation.

4. EXPERIMENTAL RESULTS

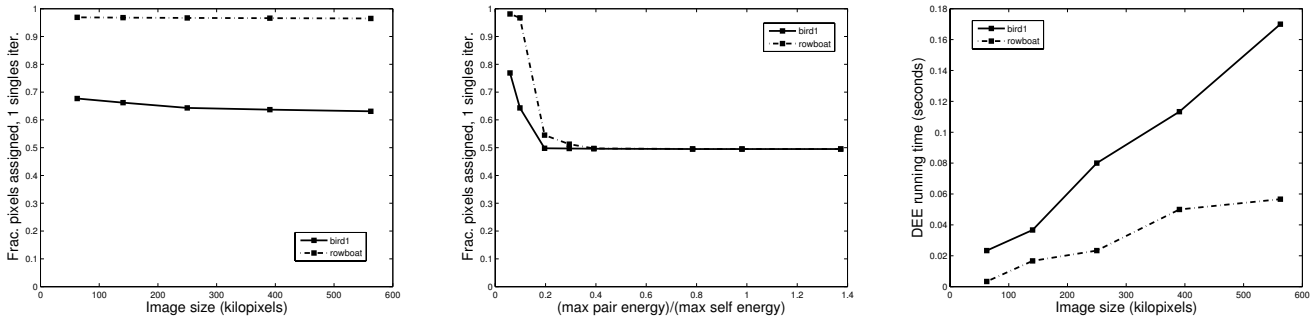
We compared four cases: (1) EK running alone, (2) BK running alone, (3) DEE followed by EK, and (4) DEE followed by BK. The four methods produced identical segmentations.

Eliminating power. For most images, DEE singles was able to assign a significant percentage of pixels after a single iteration, regardless of the quality of the final segmentation (see Figure 1). The fraction of pixels assigned, or *eliminating power* (EP), decreased slightly with image size (Figure 3a)[2], but significantly as the dominance of the pairwise interactions in the objective function increased (Equation 1). Figure 3b plots EP versus the ratio of pairwise energy and self energy terms, which was varied by changing the parameter c in Equation 2. EP dropped to $\approx 50\%$ for these images when the maximum pair energy was $\approx 40\%$ of the maximum self energy.

Running time. The running time was roughly linear in the image size (Figure 3c). We compared the running times of the four cases listed above, all running on a 3-GHz processor. In case (3), DEE singles was run until no more pixels could be assigned. Because EK itself is quite slow, we restricted our results to 50×50 pixel images (resized using MATLAB). In (4), a single iteration of DEE singles was run on 500×500 pixel images. Table 1 compares the running times (averaged over 3 trials). A small number of DEE cycles sped up the EK min-cut computation. However, these results are preliminary, as our implementations of EK and DEE have not been rigorously optimized. Nevertheless, the results look promising.

Currently, DEE+BK is slower than BK alone, even though a single iteration of DEE can assign a large fraction of pixels. This may be due to other heuristics in the standard BK implementation, as well as our currently non-optimal implementation of DEE, and we will investigate this further.

[2]In this case, the pair energies were on average kept at $\approx 10\%$ of the self energies so that, with 8 neighbors, the maximum sum over pairwise energies for a pixel was roughly equal to the maximum self energies.



(a) (b) (c)

Figure 3: Eliminating power (fraction of pixels assigned) and running time of DEE applied to two images, using the energy function shown in Eq. 2. (a) Eliminating power as a function of image size. (b) Eliminating power as a function of the dominance of the pair terms. (c) Running time (averaged over 3 trials) of DEE as a function of image size, for one DEE singles iteration, including time to make the new graph to pass onto the BK algorithm. Images were resized using MATLAB.

5. CONCLUSIONS AND FUTURE WORK

We have shown dead-end elimination to be an effective preconditioner for the Edmonds–Karp algorithm applied to the image segmentation problem, assigning a large portion of pixels quickly. We expect that DEE will improve running times for other min-cut methods, including Boykov–Kolmogorov, if we can further improve our implementation. Such improvements, along with further experimentation, are future work.

The problem of segmenting an image into more than two regions is NP-hard. Many min-cut algorithms no longer apply, but DEE can still greatly reduce the search space such that an exact solution can be found by other methods, e.g. the integer programming method of Kingsford *et al.* [10]. Evaluating the eliminating power of DEE applied to multiway cut problems is future work.

DEE allows for more flexibility than graph-based methods in choice of energy function. Energy functions need not follow the rules outlined by Kolmogorov and Zabih [11]. A more flexible energy function may result in more accurate segmentations, and applying DEE followed by combinatorial optimization methods (e.g. integer programming) might be a desirable way to compute multiway cuts.

Acknowledgments. Implementation advice from M. Altman and comments from V. Kolmogorov, S. Paris, and B. Tidor helped improve this paper. We thank D. Karger for suggesting a simplified interpretation of the heuristic and F. Durand for input images. The authors were supported by a DOE CSGF fellowship under grant number DE-FG02-97ER25308 and a National Science Foundation graduate fellowship.

6. REFERENCES

[1] Z. Wu and R. M. Leahy, "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation," *IEEE T. Pattern Anal.*, vol. 15, no. 11, 1993.

[2] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE T. Pattern Anal.*, vol. 22, no. 8, August 2000.

[3] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images," in *Proc. of IEEE ICCV*, July 2001.

[4] Y. Boykov and V. Kolmogorov, "Computing geodesics and minimal surfaces via graph cuts," in *Proc. of IEEE ICCV*, November 2003.

[5] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in computer vision," *IEEE T. Pattern Anal.*, September 2004.

[6] R. F. Goldstein, "Efficient rotamer elimination applied to protein side-chains and related spin glasses," *Biophys. J.*, May 1994.

[7] J. Desmet, M. De Maeyer, B. Hazes, and I. Lasters, "The dead-end elimination theorem and its use in protein side-chain positioning," *Nature*, vol. 356, April 1992.

[8] N. A. Pierce and E. Winfree, "Protein design is NP-hard," *Protein Engineering*, vol. 15, no. 10, 2002.

[9] J. Edmonds and R. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, no. 2, April 1972.

[10] C. L. Kingsford, B. Chazelle, and M. Singh, "Solving and analyzing side-chain positioning problems using linear and integer programming," *Bioinformatics*, vol. 21, no. 7, 2005.

[11] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?," *IEEE T. Pattern Anal.*, vol. 26, no. 2, pp. 147– 159, February 2004.