Computation Complexity of Branch-and-Bound Model Selection

Ninad Thakoor, Venkat Devarajan,
Electrical Engineering Department,
University of Texas at Arlington, USA 76010,
Email: ninad.thakoor@mavs.uta.edu, venkat@uta.edu.

Jean Gao,
Computer Science and Engineering Department,
University of Texas at Arlington, USA 76010,
Email: gao@uta.edu.

Abstract

Segmentation problems are one of the most important areas of research in computer vision. While segmentation problems are generally solved with clustering paradigms, they formulate the problem as recursive. Additionally, most approaches need the number of clusters to be known beforehand. This requirement is unreasonable for majority of the computer vision problems. This paper analyzes the model selection perspective which can overcome these limitations. Under this framework multiple hypotheses for cluster centers are generated using spatially coherent sampling. An optimal subset of these hypotheses is selected according to a model selection criterion. The selection can be carried out with a branch-and-bound procedure. The worst case complexity of any branch-and-bound algorithm is exponential. However, the average complexity of the algorithm is significantly lower. In this paper, we develop a framework for analysis of average complexity of the algorithm from the statistics of model selection costs.

1. Introduction

In the field of computer vision, solutions to various segmentation problems such as image segmentation, motion segmentation, disparity segmentation, and structure-and-motion segmentation can be expressed as clustering problems. In a clustering problem, the data points have to be assigned to various clusters by minimizing an assignment cost. To compute this cost, cluster centers should be known which can only be computed if the cluster assignments is known. This "chicken-and-egg" dilemma leads to the conventional iterative formulation for clustering. The clustering aims to optimize the assignment cost to achieve a (locally) optimal solution. If the number of clusters is increased, the cost for the same data reduces. The degenerate case for this happens when one cluster corresponds to one data point and the corresponding clustering cost is zero. Clearly, such a scenario is undesirable. Thus the clustering cost must be penalized for additional clusters. A variety of model selection methods exist which incorporate this idea. However, to apply model selection to clustering, candidate models for varying number of clusters have to be generated sequentially and the best model according to a model selection criterion can then be selected. Thus the problem of clustering and model selection becomes iterative and sequential. Additionally, general formulation of clustering is unaware of spatial relationships in the image and cannot utilize them. The iterative and sequential problem of model selection can be simplified to a one step optimization by using the knowledge that the clusters are spatially coherent. Hypotheses for cluster centers can be generated by sampling spatially coherent data points. Once the hypotheses are known, a subset of these hypotheses can be selected by optimizing a model selection criterion.

This idea is utilized in structure and motion segmentation approaches proposed recently [8, 9, 14]. Schindler and Suter [9] carry out multi-body structure and motion segmentation from two camera views. After the correspondences are established, they are grouped together based on the spatial coherence. From each group of correspondences, a hypothesis for underlying structure and motion is generated using random sample consensus (RANSAC) [6]. A geometrically robust information criterion (GRIC) [16] is optimized to select the best subset of hypotheses. The optimization is carried out with Tabu search [5]. In [8], Li solves the two-view motion segmentation problem starting from a set of candidate motions generated by applying spatial coherence, prior distribution, chirality con-

2009 IEEE 12th International Conference on Computer Vision (ICCV)
978-1-4244-4419-9/09/$25.00 ©2009 IEEE

straints etc. The segmentation problem is then formed as a facility location problem and solved with linear programming relaxation [2, 7]. In our previous work [14], we generate hypotheses for structure and motion by applying local sampling followed by nonmaximal suppression. We optimize the Bayesian information criterion (BIC) [10] with a branch-and-bound strategy.

In this paper, we analyze our framework in [14] for the multi-hypothesis branch-and-bound model selection and its average computational complexity. The average computational complexity of branch-and-bound algorithms which search over random trees has been explored by a number of researchers [4, 11, 12, 13, 17, 18]. The term "random" applies to the structure of the tree and weights of the tree edges in general. However, for the multi-hypothesis branch-and-bound model selection problem, the structure of the tree is deterministic and only the weights of the tree edges are random. Thus a separate treatment for the complexity of the problem becomes necessary.

This paper is organized as follows: Section 2 formulates the framework to estimate the expected complexity of the branch-and-bound search for the problem. Section 3 outlines the computational complexity estimation. The computation of various quantities involved in the estimation of complexity of the algorithm is discussed in section 4. Section 5 presents the results achieved for the expected complexity and conclusions are presented in section 6.

2. Branch-and-bound as an edge-weighted tree search problem

Under sampling based clustering formulation, given a set of N_h hypotheses $H = \{H_1, H_2, \ldots, H_{N_h}\}$ for cluster parameters, we have to choose a subset Θ which optimizes the model selection criterion.

$$\mathcal{C}(\Theta) = -\log(\mathcal{L}_\Theta) + \alpha \cdot K, \tag{1}$$

where α is a positive constant and $\mathcal{L}_\Theta$ gives the likelihood of the data for a model Θ.

From this cost function and the corresponding bounds, a branch-and-bound algorithm can be implemented in various ways [1]. For complexity analysis, we adapt a generic queue based tree search implementation of the branch-and-bound procedure from [11]. The solution tree for the problem can be explored using various search strategies. We list a few of these methods here[11],

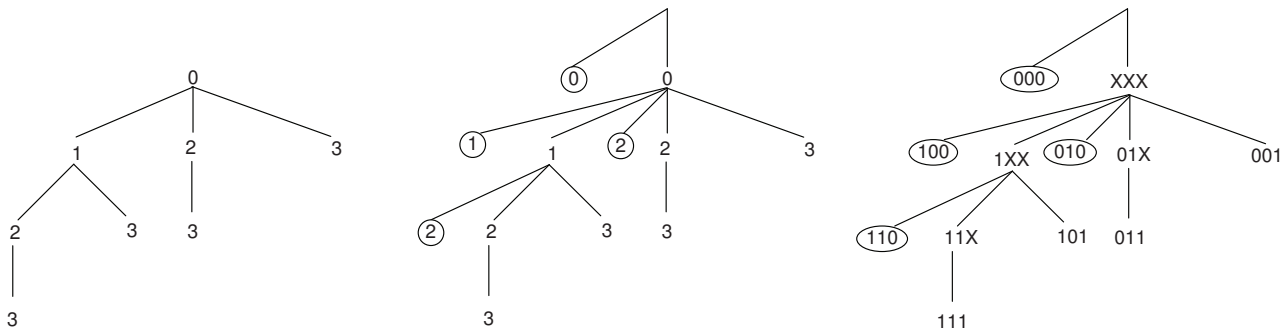
- Best bound first (BBF)
- Ordered depth first
- Generation order depth first
- Ordered breadth first
- Generation order breadth first

These methods prioritize the search of nodes in different ways. We concentrate on the BBF approach which explores the least number of nodes before it reaches the optimal solution [11]. In a BBF implementation, child nodes of the currently popped node are inserted in a queue and the minimum cost node is popped out of queue. The algorithm terminates when the first leaf node is popped out.

According to our formulation in [14], each node of the solution tree has cost $\mathcal{C}(\Theta)$ which represents a node as a solution and a lower bound on cost leading from the current node $\mathcal{C}_{\text{Lower}}(\Theta)$ which represents a node as a partial solution for the problem. This gives rise to a binomial tree [3] of order N_h as the representation of the model selection problem (see figure 1(a)). However, in a typical tree search problem only the leaf/terminal nodes can represent a solution. To incorporate this, we modify the original tree structure and add a "twin" node to each internal node of the binomial tree. The original and the updated tree structure are shown in figure 1(a) and (b) respectively. The circled nodes in (b) are the newly added twin nodes. In the updated tree, the terminal nodes (i.e., terminal nodes from the original tree and newly added twin nodes) represent the solutions and the internal nodes represent the partial solutions.

To represent each node uniquely, we devise a binary representation for each node. In this N_h bit wide representation, if a node includes a hypothesis h then $(N_h - h)$th bit of the representation is one and if it does not include the node h then $(N_h - h)$th bit is zero. For the internal nodes it also includes an additional symbol X which indicates a hypothesis that can be included in future. The $(N_h - h)$th bit is set to X if any child nodes can include the hypothesis h. One can quickly get the "twin" node of an internal node by replacing Xs with zeros.

Each edge of the updated tree has a nonzero positive weight associated with it. The cost of reaching a node can be computed by adding costs of all the edges along the path from the node to the root of the tree. For such a tree, the cost of all the child nodes is greater than the cost of their parent nodes. This agrees with the property of the branch-and-bound tree according to which the cost of any solution leading from a node is always greater than the cost of the node under consideration. The cost associated with a terminal node is the cost of the solution $\mathcal{C}(\Theta)$ associated with the terminal node. On the other hand, the cost associated with an internal node is the bound on the cost of the partial solutions represented by the internal node $\mathcal{C}_{\text{Lower}}(\Theta)$. The least cost terminal node in the edge-weighted tree corresponds to the optimal solution for the branch-and-bound process. Thus, our branch-and-bound approach can be seen as a least cost leaf search problem for the updated edge-weighted tree.



(a) (b) (c)

Figure 1. (a) Original branch-and-bound tree for $N_h = 3$, (b) Its arc-weighted equivalent, (c) Binary coding for the tree nodes

3. Average complexity

The worst case computational complexity of any branch-and-bound search algorithm is same as the complexity of the brute force search. However, a branch-and-bound approach is generally applied to an **NP** hard global optimization problem for which the worst case complexity gives a little or no insight into the performance of the approach. In such a situation, the average or expected computational complexity would give a more reasonable estimate of the performance of the approach. In this section, we formulate a framework to estimate the expected computational complexity for the branch-and-bound model selection approach based on the representation in the previous section.

The first leaf node popped from the priority queue during BBF search is the optimal solution[11]. This also means that the complexity, i.e. the number of nodes popped out before the optimal node, is same as the number of internal nodes which have their costs less than the optimal cost. Additionally, the optimal node has the cost less than all the other leaf nodes by definition.

Let T denote the set of all leaf/terminal nodes of the tree and I denote the set of all internal nodes of the tree. The optimality probability of the node i, $\Pr_o(i)$, denotes the probability that the node i is optimal, i.e. it has the least cost among the terminal nodes.

$$\Pr_o(i) = \prod_{\forall j \in T \setminus i} \Pr(W(i) < W(j)) \tag{2}$$

The cost probabilities $\Pr(W(i) < W(j))$ are probabilities of comparison of the sum of edge weights leading to nodes i and j. These can be seen as probabilities of comparison between two sums of random variables and thus are given by $\Pr_T(S_m < S_n)$. Here S_m and S_n are the sums of m and n random variables respectively ($1 \leq n \leq N_h, 1 \leq m \leq N_h$). Note that it is not necessarily true that $m = depth(i)$ and $n = depth(j)$. One has to remove the common edges along the path to the root node from the node depth to get values of m and n. If the number of common edges is l then $m = depth(i) - l$ and $n = depth(j) - l$. We define these nodes to have a relationship of order (m, n). In graph theory terms, the relationship between nodes can be seen as the simple path between them and $(m + n)$ gives the length of the simple path.

Due to the recursive structure of the tree, the weight relationships repeat themselves. Thus $\Pr_o(i)$ can be expressed as,

$$\Pr_o(i) = \prod_{m=1}^{N_h} \prod_{n=1}^{N_h} \Pr_T(S_m < S_n)^{O_i(m,n)} \tag{3}$$

Here O_i is the optimality matrix for the node i and its (m, n)th element indicates the number of times the relationship (m, n) (and hence the term $\Pr_T(S_m < S_n)$) appears in the computation of $\Pr_o(i)$.

$N(i)$ denotes the number of nodes explored if the node i is optimal. When the node i is optimal, the internal node j is explored only if its cost is less than the cost of the optimal node i. Thus the complexity when the node i is optimal is,

$$N(i) = \sum_{\forall j \in I} \Pr(W(j) < W(i)) \tag{4}$$

Similar to the optimality probability $\Pr_o(i)$, the complexity $N(i)$ of the node can be expressed as,

$$N(i) = \sum_{m=1}^{N_h} \sum_{n=1}^{N_h} \Pr_I(S_n < S_m) \cdot R_i(m, n) \tag{5}$$

Here R_i is the complexity matrix for the node i and its (m, n)th element indicates the number of times relationship (m, n) (and hence the term $\Pr_I(S_m < S_n)$) repeats in computation of $N(i)$. Note that different subscripts are used for probabilities $\Pr_T$ and $\Pr_I$, as the sums compared by these probabilities differ slightly. For $\Pr_T$, one of the weights in both sums is for an edge from an internal node to a leaf node while all the other weights are for edges between internal nodes. For $\Pr_I$, all the weights correspond to edges between internal nodes. If we assume that this difference is

negligible then,

$$\Pr(S_m < S_n) = \Pr_T(S_m < S_n) = 1 - \Pr_I(S_n < S_m). \tag{6}$$

With the optimality probability $\Pr_o(i)$ and the complexity $N(i)$, the expected complexity $\overline{N}$ can be estimated as,

$$\overline{N} = \frac{\sum_{\forall i \in T} \Pr_o(i) N(i)}{\sum_{\forall i \in T} \Pr_o(i)}. \tag{7}$$

4. Computing cost probabilities, optimality matrix and complexity matrix

The optimality matrix O_i and the complexity matrix R_i are different for nodes which do not have relationship $(1,1)$ and also change with the order of the tree. However, the probabilities $\Pr_T(S_m < S_n)$ and $\Pr_I(S_n < S_m)$ are only determined by the distribution of edge weights.

For a typical model selection problem, the distribution for the sums does not have a closed form solution or it is unknown. In such a case, a close approximation of $\Pr_T(S_n < S_m)$ and $\Pr_I(S_n < S_m)$ or $\Pr(S_n < S_m)$ can be generated with sampling.

The arc weighted tree can be seen as a binomial tree with an added "twin" node for all the internal nodes. The optimality matrix O_i is computed by comparing each leaf node i with all the other leaf nodes. Recursive properties of the binomial tree can be used in computation of O_i. To compute O_i, we transform the cost weighted tree back to the binomial tree by merging the twin nodes with the internal nodes and retaining the binary representation of the twin nodes after merging.

Computation of O_i relies on the observation that *a simple path between two nodes of a binomial tree includes the root node of only the smallest subtree including both the nodes*. Note the following important properties before proceeding to compute O_i.

- Depth of a node, i.e. $d(i)$, is equal to the number of ones in the binary representation.
- Each node i belongs to a unique combination of binomial subtrees $T_0, T_1, \ldots, T_{d(i)}$ and location of ones in the representation indicates the order of binary subtrees, e.g. the most significant bit indicates a binary subtree of order $N_h - 1$ and the least significant bit indicates a binary subtree of order 0. Note that all the nodes belong to a subtree T_0 of order N_h.
- The number of nodes belonging to a subtree T_t at depth k is given by,

$$N_t(k) = \begin{cases} \binom{T_t}{k-t}, & \text{if } 0 \le k-t \le T_t; \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

where $k = 0, 1, 2, \ldots, N_h$ and $t = 0, 1, 2, \ldots, d(i)$.

Since a node i only belongs to subtrees $T_0, T_1, \ldots, T_{d(i)}$, to compute O_i, we have to analyze these subtrees alone. The binomial tree can be split into these subtrees and can be analyzed subtree by subtree, starting with the largest subtree T_0. For each subtree, we select the nodes which exclusively belong to the subtree under consideration. This can be done by simply removing the nodes belonging to the next largest subtree from the subtree under consideration. Finally, one has to offset the result of merging of the "twin" nodes. The merging leads to relationships of order $(m, 0)$ and $(0, n)$ which would have been of the order $(m+1, 1)$ and $(1, n+1)$ otherwise. Also, we have to remove relationship $(0, 0)$ which corresponds to comparison of the node i with itself.

The algorithm to compute the optimality matrix O_i follows.

1. Initialize $T = \{T_0, T_1, \ldots, T_d\}$ = the subtree membership of the node i, $d(i)$ = depth of the node i, set $O_i(1,1) = -1$ and all the other elements of O_i equal to zero.
2. Set $t = 0$ such that current subtree $T_t = T_0$.
3. For subtree T_t, at each depth $k = 0, 1, \ldots, N_h$ compute $M_t(k)$ the number of nodes which belong exclusively to subtree tree T_t.

$$M_t(k) = \begin{cases} N_t(k) - N_{t+1}(k), & \text{if } t < d(i); \\ N_t(k), & \text{Otherwise.} \end{cases}$$

4. If $k > 0$ and $d(i) - t > 0$, set $O_i(d(i)-t, k) = O_i(d(i)-t, k) + M_t(k)$ else set $O_i(d(i)-t+1, k+1) = O_i(d(i)-t+1, k+1) + M_t(k)$.
5. Set $t = t + 1$. If $t \le d(i)$, then go to step (3), else terminate the algorithm.

To compute the complexity matrix R_i, each leaf node i has to be compared with internal nodes of the cost weighted tree only. After the "twin" node merging, one has to compare each node i of the merged tree with all the internal nodes of the tree. Note that internal nodes of the binomial tree of order N_h form a binomial tree of order $N_h - 1$. Thus similar to (8), the number of internal nodes belonging to subtree T_t at depth k is given by,

$$L_t(k) = \begin{cases} \binom{T_t - 1}{k-t}, & \text{if } 0 \le k-t \le T_t - 1; \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

With this variation the complexity matrix R_i can be calculated similar to O_i.

5. Experimental results

The multi-hypothesis branch-and-bound model selection with branch-and-bound was implemented for multiple motion and segmentation (MSaM) problem and its expected

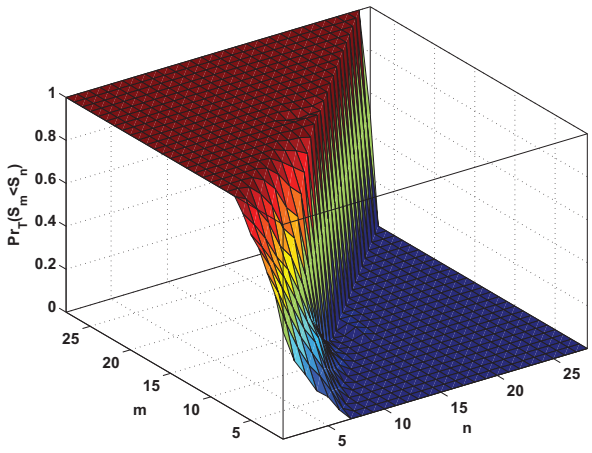


Figure 2. $\Pr_T(S_m < S_n)$ for the structure and motion segmentation problem.

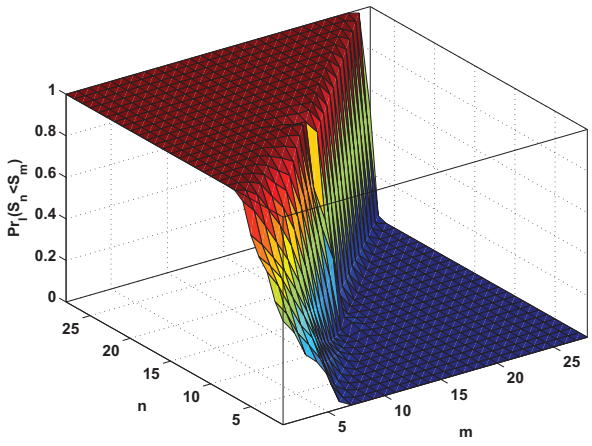


Figure 3. $\Pr_I(S_n < S_m)$ for the structure and motion segmentation problem.

computation complexity was evaluated. To generate the motion hypotheses, for each matched image feature the fundamental matrix was computed from its neighborhood. Matches in the neighborhood were used to compute the fundamental matrix using "Structure and Motion Toolkit" from [15]. Similar to RANSAC, outliers and inliers were selected for each fundamental matrix with a threshold. To avoid repeated hypothesis which are similar, non maximal suppression was carried out for the matches based on the number of inliers. Finally, the surviving hypotheses were arranged in decreasing order of the number of inliers. The Bayesian information criterion (BIC) was optimized for these hypotheses to select the optimal hypotheses combination.

The proposed branch and bound model selection approach was tested with synthetic data. For the experiments, 100 random 3D motions were generated. At a time four motions were combined together to form an experimental data sets. Also, randomly selected motion samples remaining motions were added to the data as outliers. Since the number of hypotheses N_h cannot be explicitly controlled, N_h varies as number of motion samples and their spatial configuration changes. For our experiments, we selected 50 samples per motion which gave us N_h close to 20 to 30. To estimate the probabilities $\Pr_T(S_m < S_n)$ for the MSaM problem, we randomly generated pair of hypotheses and compared their BIC values. To calculate the probabilities $\Pr_I(S_m < S_n)$, BIC value of a randomly generated hypothesis was compared to bound on BIC value of another randomly generated hypothesis. Figures 2 and 3 show the probability matrices $\Pr_T(S_m < S_n)$ and $\Pr_I(S_n < S_m)$ respectively.

Once probability matrices $\Pr_T(S_m < S_n)$ and $\Pr_I(S_n < S_m)$ are known from sampling, the complexity for the branch and bound search can be estimated by evaluating (7). Figure 4 shows the estimated expected complexity for the MSaM problem along with other tree search problems when edge weights are uniform iids and squared Gaussian iids. The worst case complexity which is equivalent to a brute force search is also shown for comparison. Clearly, the expected complexity of the branch and bound search depends on the distribution of the edge weights. This distribution is captured by probabilities $\Pr_T(S_m < S_n)$ and $\Pr_I(S_n < S_m)$.

As seen from the plots, although the expected complexity is much lesser than the worst case complexity for the branch and bound, still it is exponential for the most part. The rate of exponential depends on how quickly the off diagonal values of probability matrices $\Pr_T(S_m < S_n)$ and $\Pr_I(S_n < S_m)$ drop to near zero/ rise close to one. On the other hand, for the structure and motion problem, the increase in the complexity as $N_h > 15$ is not as drastic as $N_h < 15$. This again is a result of the off diagonal values of probability matrices $\Pr_T(S_m < S_n)$ and $\Pr_I(S_n < S_m)$ almost all of which drop to near zero/ rise close to one for $N_h > 15$.

Figure 5 compares the estimated expected complexity of the problem with the experimentally observed complexity of the problem. We ran 400 experiments with different data sets to find the number of nodes explored before optimal solution was found. These experiments were then separated based on value N_h and sorted according to increasing complexity. The lengths of plots were normalized horizontally to 100 for easy comparison of complexity for various values of N_h. As seen in Figure 5, although the expected complexity is slightly overestimated, it still provides a satisfactory estimate for the expected complexity.

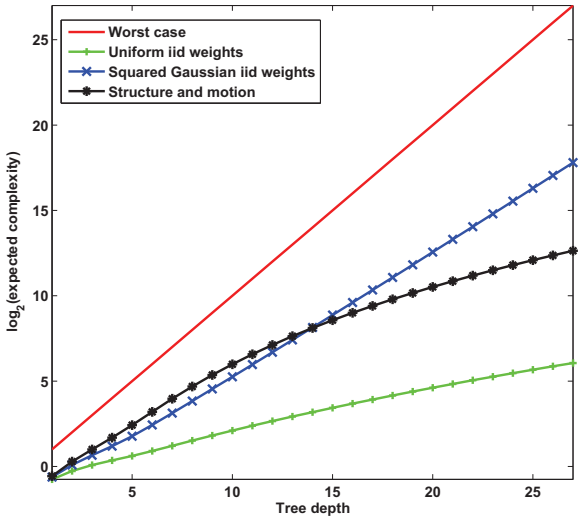


Figure 4. Expected complexity.

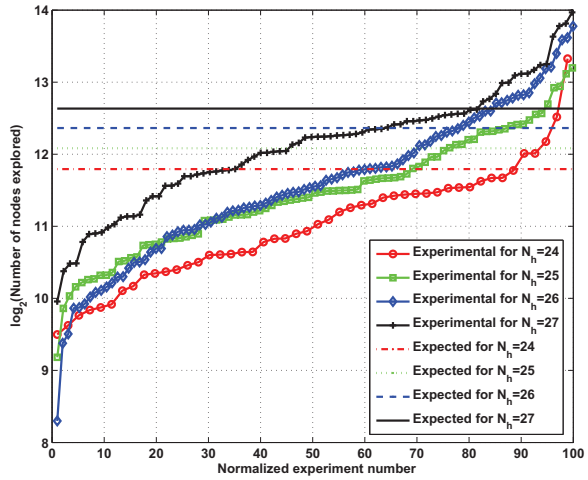


Figure 5. Comparison of expected and actual complexity.

6. Conclusion

In this paper, we analyzed the average complexity of a branch-and-bound algorithm for model selection in computer vision problems. It can be seen from the experiments that the average complexity of the algorithm is than the worst case complexity. Thus branch-and-bound based model selection algorithms are practical for hypothesis selection process which has moderate number of hypothesis and when size of optimal subset is small. With problem specific bounds and/or added heuristics, the computational complexity of the branch-and-bound algorithm can be improved further.

References

[1] M. Brusco and S. Stahl. *Branch-and-Bound Applications in Combinatorial Data Analysis*. Springer, 2005. 2

[2] F. A. Chudak and D. B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput.*, 33(1):1–25, 2004. 2

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, USA, 2001. 2

[4] L. Devroye and C. Zamora-Cura. On the complexity of branch-and bound search for random trees. *Random Struct. Algorithms*, 14(4):309–327, 1999. 2

[5] F. Glover and M. Laguna. *Modern heuristic techniques for combinatorial problems*, chapter Tabu search, pages 70–150. John Wiley & Sons, Inc., 1993. 1

[6] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, New York, NY, USA, 2000. 1

[7] D. S. Hochbaum, editor. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., Boston, MA, USA, 1997. 2

[8] H. Li. Two-view motion segmentation from linear programming relaxation. In *Proceedings of CVPR*, pages 1–8, June 2007. 1

[9] K. Schindler and D. Suter. Two-view multibody structure-and-motion with outliers through model selection. *IEEE Trans. Patt. Anal. Mach. Intell.*, 28(6):983–995, 2006. 1

[10] G. Schwarz. Estimating the dimension of a model. *Ann. Statist.*, 6(2):461–464, 1978. 2

[11] D. R. Smith. *On the Computational Complexity of Branch and Bound Search Strategies*. PhD thesis, Duke University, 1979. 2, 3

[12] D. R. Smith. Random trees and the analysis of branch and bound procedures. *J. ACM*, 31(1):163–188, 1984. 2

[13] H. S. Stone and P. Sipala. The average complexity of depth-first search with backtracking and cutoff. *IBM J. Res. Dev.*, 30(3):242–258, 1986. 2

[14] N. Thakoor and J. Gao. Branch-and-bound hypothesis selection for two-view multiple structure and motion segmentation. *Proceedings of CVPR*, pages 1–6, June 2008. 1, 2

[15] P. Torr. *A Structure and Motion Toolkit in Matlab*. http://cms.brookes.ac.uk/staff/PhilipTorr/Beta/torrsam.zip. 5

[16] P. H. S. Torr. Geometric motion segmentation and model selection. *Phil. Trans. R. Soc. A*, 356(1740):1321–1340, May 1998. 1

[17] W. Zhang. Branch-and-bound search algorithms and their computational complexity. Technical Report ISI/RR-96-443, University of southern California / Information sciences institute, May 1996. 2

[18] W. Zhang and R. E. Korf. Performance of linear-space search algorithms. *Artificial Intelligence*, 79:241–292, 1995. 2