アルゴリズム 74−5
(2000. 9. 21)

多変数パラメトリック探索による最小値最大最適化問題の解法

徳山 豪
東北大学大学院情報科学研究科

概要

ある最適化問題があったときに、その入力にパラメタを付加して，最適化問題の解が最も良くなる」パラメタ値を求めるという問題は制御解析などでは基本的な問題である。特に１パラメタ問題ではパラメトリック探索という手法が確立しており，元の最適化問題が効率よい並列アルゴリズムを持つときには１パラメタ問題の（直列計算での）強多項式アルゴリズムが得られる。本論文では定数個（ｄ個）のパラメタを持つパラメトリック最大最小化（もしくは最小最大化）問題でｄ次元版のパラメトリック探索で効率よく解ける物を考える。典型的な問題としては: (1) パラメトリック最大最小全域木問題やパラメトリックマトロイド基底問題, (2) 計算幾何学の諸問題 (3) グラフの最小カット最大化などが挙げられる。

Max-Min parametric optimization problems and multi-dimensional parametric searching

Takeshi Tokuyama
GSIS, Tohoku University, tokuyama@dais.is.tohoku.ac.jp

概要

The parametric MaxMin problem, which finds the parameter values maximizing the weight of a solution of a combinatorial minimization problem, is a fundamental problem in sensitivity analysis. The *parametric search* paradigm gives an efficient sequential algorithm if we have only one parameter and the original non-parametric problem has an efficient parallel algorithm. We consider the parametric MaxMin problem with d parameters for a constant d, and solve it by using d-dimensional version of the parametric search paradigm. Typical results obtained as applications are: (1) $O(n \log^d n)$ time algorithm to compute a MaxMin spanning tree of a graph with n edges, and (2) efficient solutions of geometric matching problems and minimum diameter bridging problems, and (3) $O(n^2 \log^{4d+1} n)$ time algorithm to compute a MaxMin cut of a graph with n vertices.

1 Introduction

Consider an optimization problem $\boldsymbol{Q}$ that finds a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ minimizing an objective function

$$z = \textstyle\sum_{i=1}^{n} c_i x_i$$

under given constraints on $\boldsymbol{x}$.

Suppose that we have a set of d real parameters $\{\lambda_1, \lambda_2, \ldots, \lambda_d\}$ for a constant d. If we replace the coefficients c_i of the objective function with a concave function $c_i(\lambda)$ on the parameter vector $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_d)$ for each $i = 1, 2, \ldots, n$, we obtain a d-dimensional parametric version $\boldsymbol{Q}(\lambda)$ of the optimization problem $\boldsymbol{Q}$.

For each fixed parameter vector λ, let $z(\lambda) = \min_{\boldsymbol{x}} \sum_{i=1}^{n} c_i(\lambda) x_i$, where the minimum is taken over all vectors satisfying the constraints. Then, $z(\lambda)$ becomes a concave function in the parameters, and its locus $\mathbf{L}(\lambda) : z = z(\lambda)$ becomes a concave hypersurface in the (λ, z)-space, which has $d+1$ dimensions. Let R be a (possibly unbounded) convex polygonal region in the d-dimensional space such that λ should be contained. We call R the *feasible region* of the parameters (note that it is different from the feasible region of $\boldsymbol{x}$). We assume that R is a convex polytope with M facets. Number of lower dimensional faces of R is irrelevant to our analysis.

The problem of finding the value $\lambda \in R$ maximizing $z(\lambda)$ is called a *MaxMin parametric optimization* problem. The MaxMin parametric optimization problem is a fundamental problem in sensitivity analysis [9]. For example, suppose we have a weighted undirected graph G, and the original problem $\boldsymbol{Q}$ is the problem of finding the minimum cut of the graph. If the weight of edges may depend on a set of parameters, the minimum cut depends on the parameters, and it is a natural problem to find a parameter vector to maximize the minimum cut value. We can also deal with the MinMax problem for a problem with convex weights if we replace $c_i(\lambda)$ with $-c_i(\lambda)$ for each i.

The MaxMin parametric optimization problem with one parameter is well-studied in the literature. Suppose that we have an algorithm $\mathcal{D}$ to solve the original problem $\boldsymbol{Q}$ in $T_{\mathcal{D}}$ time. Hence, we can compute $z(\lambda)$ together with the associated solution vector $\boldsymbol{x}$ for a fixed λ in $T_{\mathcal{D}}$ time. Since $z(\lambda)$ is a concave function, we can apply the binary search paradigm to solve the MaxMin parametric optimization problem. The time complexity of the above binary search method depends on the precision of the problem, in other words, it takes $O(T_{\mathcal{D}} \log(n\Gamma))$ time if the description of the constraints and the coefficients of $c_i(\lambda)$ are given by using quotient numbers of $O(\log \Gamma)$-bit integers.

It is desired to design an efficient algorithm whose running time is independent of Γ (on a model counting comparisons and arithmetic operations). If we have a set (with a size independent of Γ) of candidate values of the parameter such that the set contains the optimal value λ_{opt}, we can perform the binary search on it. Although the set of all break points of the piecewise linear function $z(\lambda)$ is such a set, it is expensive to compute it. Therefore, it is desired to solve the parametric MaxMin problem by performing the binary search without computing a large size of candidate set. Megiddo's parametric search [16, 19] is a well-known paradigm to realize this idea.

For $d \geq 2$, the MaxMin problem of the *minimum element finding problem*, which finds the minimum of n real numbers, is simply a linear programming (LP) problem (if the weight functions are linear), and it can be solved in linear time for any fixed d [17]. The fixed dimensional LP has been generalized and studied from several aspects (see [6]). However, it seems that efficient solutions of other MaxMin parametric optimizations with more than one parameters are not well-studied in the literature.

The aim of this paper is to apply the parametric search paradigm to the d-dimensional MaxMin parametric optimization problems. The binary search method can be applied, and the running time is $O(T_{\mathcal{D}}(\log n\Gamma)^d)$. We would like to remove the influence of Γ from the time complexity. In a multi-dimensional case, the locus $\mathbf{L}(\lambda)$ becomes a convex polytope in $(d+1)$-dimensional space, and its construction will become expensive and complicated.

The multi-dimensional parametric search paradigm was given by Cohen and Meggido [5] to solve the parametric cycle detection problem. Matoušek and Schwarzkopf [15] gave a geometric interpretation, applied it to give an efficient LP-query for constant-dimensiomal LP problems, and predicted that the paradigm would be useful for solving problems in compuational geometry. We apply the multi-dimensional parametric search paradigm to the MaxMin parametric optimization problem and obtain the following general framework:

Theorem 1.1 *Consider an optimization problem $\boldsymbol{Q}$ that can be solved in $T_{\mathcal{D}}$ sequential time, and also has a generic parallel algorithm with $O(\tau)$ parallel time on $O(N)$ pocessors by using linear decision operations with respect to the parameters. Here, a linear decision is to decide whether the value of a* *linear form on parameters is positive or not. Then, the corresponding parametric MaxMin problem with d parameters can be solved in $O((N + M + T_{\mathcal{D}} \log(N + M))\tau^d(\log(N + M))^{d-1})$ sequential time, where M is the number of facets in R.*

Typically, we consider the case where each $c_i(\lambda)$ is a concave linear function with a constant number of facets (linear pieces); however, we can also apply the method for the case $c_i(\lambda)$ are concave piecewise differentiable functions, whereas the decisions in the guide algoirthm of the parametric serach are linear in the parameter.

Our general framework has several applications including polymatroid optimization, geometric optimization (in reply to Matoušek and Schwarzkoph's prediction), and graph algorithms.

2 Multidimensional parametric search paradigm for MaxMin optimization

2.1 Basic paradigm

Multidimensional parametric search uses an algorithm $\mathcal{G}$ called the *guide algorithm*, which can solve $Q(\lambda)$ for any given fixed value of λ in $O(T_{\mathcal{G}})$ time. The guide algorithm $\mathcal{G}$ is generic, i.e., it has a decision circuit (directed acyclic graph) whose graph structure is independent of the value of c_i $(i = 1, 2, \ldots n)$. Each node requiring a decision has two outgoing branches. Arithmetic computations are additions and subtractions of linear forms in c_i $(i = 1, 2, \ldots, n)$ and multiplications by constants. Hence, in the parametric situation, they are operations in the additive module of linear forms in λ. The decision of branches at each node v of the circuit is done by comparing the value $f_v(\lambda)$ of a linear function f_v with 0. (In precise, the function f_v may depend on the results of preceding decisions and arithmetic operations).

We simulate the behavior of $\mathcal{G}$ as if it had been computing $z(\lambda_{opt})$, where λ_{opt} is the parameter vector at the solution of the parametric MaxMin problem. For doing it, we use the *parametric linear test*, which answers that which side of a given hyperplane H in the parameter space the optimal value λ_{opt} is located.

We start guide algorithm $\mathcal{G}$; however, since the value λ_{opt} is not input, the algorithm halts. When the algorithm $\mathcal{G}$ halts at each node v of the decision circuit requiring a decision depending on the parameter vector λ, we want to know whether $f_v(\lambda_{opt}) \geq 0$ or $f_v(\lambda_{opt}) < 0$. Since we know the function f_v itself, we have the hyperplane H_v defined by the equation $f_v(\lambda) = 0$. The parametric linear test gives the correct decision, and hence we can proceed to the next step. If the parametric linear test needs $O(T_{\mathcal{T}})$ time, the above method computes λ_{opt} and $z(\lambda_{opt})$ in $O(T_{\mathcal{T}} T_{\mathcal{G}})$ time.

2.2 How to implement the parametric linear test

For simplicity, we assume that $M = O(N)$ and ignore M in the analysis. Let λ_{opt} be the parameter vector at which $z(\lambda)$ takes the maximum. A parametric linear test can be implemented as follows: We restrict the feasible region R on the hyparplane H_v. The restricted problem is a (d-1)-dimensional parametric problem, and its MaxMin solution can be obtained by using the algorithm (denoted by $\mathcal{D}(d-1)$) for the (d-1)-dimensional parametric problem. The maximum of $z(\lambda)$ on the hyperplane H_v is attained at the point p associated with the solution. If the weight functions are piecewise linear, we have the point p and d solution vectors $\boldsymbol{x}(e_1), \boldsymbol{x}(e_2), \ldots, \boldsymbol{x}(e')$ corresponding to linear pieces meeting at p in the slice $\mathbf{L}(\lambda) \cup H_v$. These solution vectors give d facets of $\mathbf{L}(\lambda)$ adjacent to each other. The ridge between these two facets is on a line l in the space. By definition, l contains the point p attaining the maximum of $z(\lambda)$ on the hyperplane H_v. Let H_v^+ be a halfplane defined by $f_v(\lambda) > 0$. We take any point q of $l \cup H_v^+$, and suppose that the z-value of q is not larger than that of p. In this case, we decide that $f_v(\lambda) \leq 0$ for the optimal value λ_{opt}. If the equality $f_v(\lambda_{opt}) = 0$ holds, we have already found the solution, and hence we give the decision that $f_v(\lambda_{opt}) < 0$ for continuing the process. If the z-value of q is larger than that of p, we decide that $f_v(\lambda_{opt}) > 0$. If some pieces of weight functions are nonlinear (but differentiable), it may happen that the solution vector at p is determined less than d facets. In that case, we consider a tangent plane at p to the non-linear solution vector, regard it as a degenerate case of some facets, and take any line l on its intersection with linear facets.

Therefore, the parametric search without utilizing a parallel structure works for the d-dimensional case, in which each decision (parametric linear test) is done by calling the $(d-1)$-dimensional parametric algorithm $\mathcal{D}(d-1)$. This method takes $O(T_{\mathcal{G}} T_{\mathcal{D}(d-1)}) = O((T_{\mathcal{G}})^{d-1} T_{\mathcal{D}(1)})$ time to compute the optimal value λ_{opt}.

2.3 Utilizing a parallel structure

In order to improve the time complexity, it is advantageous to use a guide algoirthm $\mathcal{G}$ that has a parallel structure such that it runs in $O(\tau)$ parallel time by using N processors (in PRAM or the Variant's comparison model). The algorithm $\mathcal{G}$ has $O(\tau)$ layers and visits at most N nodes at each layer (i.e. it runs in $O(\tau)$ parallel time using N processors). At each layer, instead of calling the parametric linear test at each visited node, we consider the arrangement $\mathcal{A}$ of hyperplanes constructed from the set $\mathcal{H}$ of all threshold hyperplans in the layer, and seek for the cell containing λ_{opt}. Precisely speaking, we add the hyperplanes defining the facet of R to $\mathcal{H}$ and form the arrangment $\mathcal{A}$.

Suppose that we can find the cell C that contains λ_{opt} of $\mathcal{A}$. If λ_{opt} is on the boundary of the cell, we have reduced the problem to a $(d-1)$-dimensional problem, and we can solve it by using $\mathcal{D}(d-1)$. Otherwise, for each v, if C is in the halfspace defined by $f_v > 0$, we can decide that $f_v > 0$ holds for λ_{opt}. Hence, we know all correct decisions at the nodes in the layer, and can proceed to the next layer.

We introduce the multidimensional prune-and-search method to search for the sell developed by Matoušek and Schwarzkopf [15]. We have a set $\mathcal{H}$ of N hyperplanes in the parameter space $\mathbf{R}^d$, and an unknown point λ_{opt} in the space. We want to construct a planar subdivision Ξ into a constant number of simplices such that the number of hyperplanes in $\mathcal{H}$ intersecting the simplex $\Delta(\lambda_{opt})$ of Ξ containing p is less than $n/2$ (with high probability by using a randomized algorithm).

This can be attained by using an ϵ-cutting, which is frequently used in computational geometry, especially for designing algorithms based on geometric divide-and-conquer. An ϵ-cutting of an arrangement $\mathcal{A}$ is a subdivision of the d-dimensional space $\mathbf{R}^d$ into d-dimensional simplices such that $O(\epsilon N)$ hyperplanes intersect (the interior of) each simplex. We apply a popular randomized method to construct a cutting (see Mulmuley [18]): We first randomly sample r hyperplanes among N hyperplanes, and construct the arrangement of them. Then, triangulate each cell into simplexes by using the *canonical triangulation.* The obtained triangulation Ξ has $O(r^d)$ simplices, and is an $O(r/\log r)$-cutting with a high probability. Theoretically better cuttings are known [2, 18], although the above method is sufficient for our use, since if we fix any unknown point p in the space, the simplex $\Delta(p)$ containing p in the cutting Ξ intersects at most cN/r hyperplanes for a suitable constant c. with a high probability if N is larger than a suitable constant, since it suffices to consider the expected number of the intersecting hyperplanes with each edge of $\Delta(p)$. The time for the construction is $O(Nr^{d-1})$. Note that we do not need parametric linear tests for the construction of the cutting.

We choose r to be a constant satisfying that $r > 2c$. Once we have the cutting, we can find a simplex $\Delta(\lambda_{opt})$ by using an $O(r^d)$ number of linear tests. We have N' hyperplanes intersecting $\Delta(\lambda_{opt})$. Since $r > 2c$, $N' \leq cN/r < N/2$ with a high probability. Now, except those corresponding to the N' hyperplanes, we know all the $N - N'$ decisions of the layer of the guide algorithm.

In order to determine the remaining decisions, instead of the original arrangement, we next consider the arrangement of N' hyperplanes intersecting Δ together with the $d+1$ bounding hyperplanes (some of them may be newly created hyperplanes) of Δ, and find its cell containing λ_{opt} by applying the same method. Thus, we can determine all the decisions in a layer of the guide algorithm with an $O(\log N)$ number of parametric linear tests. The computation time for constructing the cuttings is $O(N)$ in total for each layer.

Therefore, the time complexity of this version of parametric search follows the formula:

$$T_{\mathcal{D}(d)} = O(\tau(T_{\mathcal{D}(d-1)} \log N + N)).$$

Hence, $T_{\mathcal{D}(d)} = O((\tau \log N)^d T_{\mathcal{D}} + \tau^d (\log N)^{d-1} N)$, where $T_{\mathcal{D}}$ is the time to solve the non-parametric problem. If $M >> N$, we should replace N with $N + M$ in the formula.

3 Applications

3.1 Parametric Matroid and Polymatroid Optimization

Let us consider a *polymatroid*, which is a pair (E, r) of a set $E = \{1, 2, .., n\}$ called the *underlying set* and the *rank function* r, which assigns a nonnegative real number $r(X)$ to each $X \in 2^E$ and satisfies the monotonicit and submodular conditions. For each $i \in E$, we define a weight w_i, and consider the base $\boldsymbol{v}$ which minimizes $\sum_{i=1}^{n} w_i v_i$ under the condition that $\sum_{i \in X} v_i \leq r(X)$ and $\sum_{i \in E} v_i = r(E)$. This base is called the *minimum weight base*. See [1, 8] for the optimization theory of matroids and polymatroids.

If we replace each w_i with a linear function $w_i(\lambda)$ on d parameters, we obtain a parametric polymatroid [7, 13]. We also permit piecewise linear weight functions. The MaxMin parametric optimization is to find the parameter vector in a given feasible region R maximizing the weight of the minimum weight base.

A basic fact on a polymatroid is that the minimum weight base at λ only depends on the sorting order of the weights $w_i(\lambda)$. Therefore, we can use a parallel sorting algorithm as the guide algorithm $\mathcal{G}$ instead of an algorithm actually compute the minimum weight base This idea was proposed in Megiddo's paper [16] for solving a parametric problem on the minimum spanning tree.

Since $O(n)$ processors and $O(\log n)$ parallel time algorithm is known for sorting n elements, the result in the previous section implies that the parameter value maximizing the minimum weight of a base of a parametric polymatroid with d linear parameters can be solved in $O(T \log^{2d} n)$ time, where T is the time to compute a minimum weight base for the non-parametric version of the polymatroid. We can further improve it by using Cole [4]'s approach:

Theorem 3.1 *If each weight function is a piecewise linear concave function in d parameters with a constant number of linear pieces, the parameter values maximizing the minimum weight of a base of a parametric polymatroid can be computed in $O(T \log^d n)$ time, where T is the time to compute a minimum weight base for the non-parametric version of the polymatroid.*

A typical example of a polymatroid (indeed, a matroid) is the graphic matroid on a connected graph, in which a minimum weight base is a minimum spanning tree of the graph.

Corollary 3.2 *The parameter vector $\lambda \in R \subset \mathcal{R}^d$ maximizing the weight of the parametric minimum spanning tree of a connected graph G with n edges can be computed in $O(n \log^d n)$ time if each edge weight is a concave linear function with a constant number of linear pieces on the d parameters.*

3.2 Geometric Applications

3.2.1 Minimum diameter bridging problem

Given two convex regions P and Q in the d-dimensional Euclidean space. Suppose that P and Q have n vertices and M facets in total. We assume that both the vertex set and facet set are given. The minimum diameter bridging problem is to construct a bridge (p, q) between $p \in P$ and $q \in Q$, such that the diameter $d(p,q) + \max_{v \in V(P)} d(v,p) + \max_{w \in V(Q)} d(q,w)$ of the union of P, Q, and the bridge is minimized, where $d(x,y)$ is the Euclidean distance, and $V(P)$ and $V(Q)$ are vertex sets of P and Q, respectively. The problem has been considered for $d = 2$ and $d = 3$ in the literature, and a linear time algorithm is given for $d = 2$ [14] and a quadratic time algorithm is given for $d = 3$ [20]. Some approximation algorithms for the higher dimensional cases are considered by Chong [3]. The minimum diameter bridging problem is a MinMax optimization problem if we consider the coordinates of p and q as parameters. The feasible region of the parameter is the direct product of P and Q in the $2d$-dimensional space. The number of facets in this direct product is $2M$ although it has many lower dimensional faces.

The Euclidean distance is a convex function, and hence our objective function is a convex function. If the parameter values are given, we can compute the diameter by computing $\max_{v \in V(P)} d(v,p)$ and $\max_{w \in V(Q)} d(q,w)$ first, and then compute the summation of three distances.

We need nonlinear operations to compute the summation of three distances. However, fortunately, all decisions are linear with respect to the parameters; indeed, implicitly $\max_{v \in V(P)} d(v,p)$ is determined by the cell of the furthest Voronoi diagram of $V(P)$ containing p. Since computing the furthest Voronoi diagram is expensive, we do not prepare it, and naively compute $\{d(v,p) : v \in V(P)\}$ and compute its maximum. The probelm is basically a maximum finding problem.

Although a parallel algorithm with with an $O(\log \log n)$ time complexity by using $O(n)$ processors is known for the maximum finding, we use a slower $O(\log n)$ time $O(n)$ processor algorithm in order to apply Cole's method. Therefore, we can apply our framework.

Theorem 3.3 *The minimum diameter bridging problem can be solved in $O((n+M) \log^{2d}(n+M))$ time.*

The minimum diameter bridging problem can be extended to the following parametric geometric network-base location problem: We have sets $V_1, V_2, \ldots, V_k$ of points and a set of convex regions $P_1, P_2, \ldots, P_k$ for a constant k. Choose $p_i \in P_i$ for each $i = 1, 2, .., k$ such that $l(MST(p_1, p_2, \ldots, p_k)) + \sum_{i=1}^{k} \max_{v \in V_k} d(p_k, v)$ is minimized. Here, $l(MST(p_1, p_2, \ldots, p_k)$ is the sum of lenghts of edges in the minimum spanning tree of $p_1, p_2 \cdot \ldots, p_k$.

The parametric minimum spanning tree computation is the MinMin problem, and hence cannot be handled in our framework. Hence, instead of MST, we fix a tree topology T and consider the problem of minimizing $l(T(p_1, p_2, \dots, p_k)) + \sum_{i=1}^{k} \max_{v \in V_k} d(p_k, v)$. By applying our framework, we can solve the problem in $O((n+M)\log^{dk} n)$ time, where n is the sum of numbers of vertices in V_i $(i = 1, 2, \dots, k)$ and M is sum of the numbers of facets in P_i $(i = 1, 2, \dots, k)$. If we try the above algorithm for all possible k^{k-2} tree topologies, we can solve the parametric geometric network-base location problem.

Theorem 3.4 *The parametric geometric network-base location problem can be solved in* $O((n+M)\log^{dk}(n+M))$ *time.*

3.2.2 MinMax optimized point set matching

Given k point sets $V_1, V_2, \dots, V_k$ in a d-dimensional space, we translate V_i with a translation vector t_i to have a point set $V_i + t_i = \{v + t_i : v \in V_i\}$. and consider the union $V((\mathbf{t})) = \bigcup_{i=1,2,\dots,k}(V_i + t_i)$, where $\mathbf{t}$ is a dk-dimensional vector obtained as the direct product of t_i $(i = 1, 2, \dots, k)$. We define a estimation function F on a point set and would like to find the parameter value $\mathbf{t}$ minimizing $F(V(\mathbf{t}))$. If $F(V(\mathbf{t}))$ is a convex function, there is a possibility that we can apply multidimensional parametric searching. We can observe that we can fix $t_1 = 0$ if F is translation invariant, which holds for all applications dealed in this subsection.

A trivial case is that F is the diameter of $V(\mathbf{t})$, where it suffices to translate the center of minimum enclosing ball of each V_i to the origin. A more interesting example is the following geometric fitting problem: For each pair V_i and V_j, a one-to-one matching $\pi_{i,j} : V_i \to V_j$ between them is given. Let $dist(V_i, V_j; \mathbf{t}) = \max_{v \in V_i} d(v + t_i, \pi_{i,j}(v) + t_j)$, and let $F(V(\mathbf{t})) = \sum_{i,j} dist(V_i, V_j; \mathbf{t})$. This problem was proposed by Imai et al.[10] for the case where $k = 2$ and $\pi_{i,j}$ is the inverse of $\pi_{j,i}$ (they further consider the rotation of point set). If $k = 2$, the problem is a LP problem, and solved in linear time[10].

For $k > 3$, $F(V(\mathbf{t})) = \sum_{i,j} dist(V_i, V_j; \mathbf{t})$ is a convex function, and we can design a $O(\log n)$ time $O(n)$ processor guide algorithm that uses linear decisions ; Thus, the problem can be solved in $O(n \log^{dk} n)$ time. We remark that if we consider $F(V(\mathbf{t})) = \max_{i,j} dist(V_i, V_j; \mathbf{t})$, we need nonlinear decisions in the guide algorithm, and it fails to apply our method.

We can further modify the problem as follows: instead of one-to-one matching $\pi_{i,j}$, we have a one-to-many matching $\pi_{i,j} : V_i \to 2^{V_j}$ so that $\pi_{i,j}$ is a set (possibly an empty set) of points in V_j. We consider Let $dist'(V_i, V_j; \mathbf{t}) = \max_{v \in V_i}\{\max_{w \in \pi_{i,j}(v)} d(v + t_i, w + t_j)\}$, and let $F(V(\mathbf{t})) = \sum_{i,j} dist'(V_i, V_j; \mathbf{t})$. The solution is almost similar. Naively, the multidimensional parametric search method needs $O((\sum_{i=1}^{k} \sum_{v \in P_i} |\pi_{i,j}(v)|) \log^{dk} n)$ time.

If we use a convex linear distance instead of the Euclidean distance, some more objective functions are suitable fo the parametric searching paradigm. The sum of length of edges of the maximum spanning tree of $V(\mathbf{t})$ and the sum of m-largest distances of $V(\mathbf{t})$ are such examples. Indeed, these are parametric matroid problems, and the time complexity to solve each of these problems depends on the efficiency of the decision algorithm.

3.3 Other applications

Basically, the multi-dimensional parametric search works for the parametric MaxMin problem of any optimization problem with an efficient parallel algorithm. The minimum cut problem of a weighted graph is a typical example:

Proposition 3.5 *Given a graph G on n vertices where each edge has a piecewise linear concave parametric weight with a constant number of linear pieces on d parameters. Then, the parameter vector (in a give feasible region R) maximizing the weight of a minimum cut is computed in $O(n^2 \log^{4d+1} n)$ time.*

Proof The minimum cut problem can be solved in $O(\log^3 n)$ randomized parallel time using $O(n^2/\log^2 n)$ processors [11], and the algorithm is generic. Hence, we have $T(\mathcal{D}(d)) = O(\log^3 n(\log n T(\mathcal{D}(d-1)) + n^2/\log^2 n))$, which leads to $T(\mathcal{D}(d)) = O(n^2 \log^{4d+1} n)$. □

参考文献

[1] R. Bixby and W. Cunningham, Matroid Optimization and Algorithms, Chapter 11 of *Handbook of Combinatorics*, 551-609, Elsevier, 1995.

[2] B. Chazelle, Cutting hyperplanes for divide-and-conquer, *Discrete & Computational Geometry* **9** (1993) 145–158.

[3] Otfried Chong, private communication.

[4] R. Cole, Slowing down sorting network to obtain faster sorting algorithms, *J. ACM* **34** (1987) 200–208.

[5] E. Cohen and N. Meggido, Strongly polynomial-time and NC algorithms for detecting cycles in dynamic graphs, *in Proceedings of 21st ACM Symposium on Theory of Computing* (1989) 523–534.

[6] M. Deyer and N. Meggido, Linear programming in low dimensions, Section 38 of *Handbook of Discrete and Computational Geometry* (1997) 699–710, CRC Press.

[7] D. Eppstein, Geometric lower bounds for parametric matroid optimization problems, *Proc. 27th ACM STOC* (1996) 662–671.

[8] S. Fujishige, *Submodular Functions and Optimization*, Annals of Discrete Mathematics, 47, North Holland, 1991.

[9] D. Gusfield, *Sensitivity Analysis for Combinatorial Optimization,* Memorandum No. UCB/ERL M80/22, U.C. Berkeley, 1980.

[10] K. Imai, S. Sumino, and H. Imai, Minimax geometric fitting of two corresponding set of points, *Proc. 5th ACM Symposium on Computational Geometry* (1989) 266–275.

[11] D. Karger, Minimum cuts in near-linear time, *Proc. 28th ACM STOC* (1996) 56–63.

[12] D. Karger, P. Klein, and R. Tarjan, A randomized linear time algorithm to find minimum spanning trees, *J. ACM* **42** (1995) 321-328.

[13] N. Katoh, H. Tamaki, and T. Tokuyama, Parametric Polymatroid Optimization and Its Geometric Applications, *Proc. 10th ACM-SIAM SODA* (1999) 517–526.

[14] S. K. Kim and C. S. Shin, Computing the optimal bridge between two polygons, *HKUST Research Report TCSC-99-14* (1999) to appear in IPL.

[15] J. Matoušek and O. Schwarzkopf, Linear optimization queries, *Proceedings of 8th ACM Symposium on Computational Geometry* (1992) 16–25.

[16] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. ACM* **30** (1983) 852–865.

[17] N. Megiddo, Linear programming in linear time when dimension is small, *SIAM J. Comput.*, **12** (1984) 114–127.

[18] K. Mulmuley, *Computational Geometry, an Introduction Through Randomized Algorithms*, Princeton Hall, 1994.

[19] J. Salowe, Parametric Search, Section 37 of *Handbook of Discrete and Computational Geometry* (1997) 683–695, CRC Press.

[20] X. H. Tan, On optimal bridges between two convex regions, *Proceedings of 5th Japan-Korea Workshop on Algoirthms and Computation* (2000) 57–63.