An FPTAS for Minimizing a Class of Low-Rank Quasi-Concave Functions over a Convex Set *

Vineet Goyal
Industrial Engineering and Operations Research
Columbia University
Email: vgoyal@ieor.columbia.edu

R. Ravi
Tepper School of Business,
Carnegie Mellon University
Email: ravi@cmu.edu

Abstract

We consider the problem of minimizing a class of quasi-concave functions over a convex set. Quasi-concave functions are a generalization of concave functions and thus, NP-hard to minimize over a convex set in general. We present a simple fully polynomial time approximation scheme (FPTAS) for minimizing a fairly general class of *low-rank* quasi-concave functions. Our algorithm is based on solving a polynomial number of linear minimization problems over a convex set and computes a near-optimal solution that is an extreme point of the convex set. Therefore, it applies directly to combinatorial 0-1 problems for which the convex hull of feasible solutions is known, such as shortest paths, spanning trees and matchings in undirected graphs.

Key words: Quasi-concave programming; non-linear programming; non-convex programming; polynomial approximation schemes.

1 Introduction

In this paper, we consider the problem of minimizing a class of *low-rank* quasi-concave functions over a convex set. Quasi-concave functions are a generalization of concave functions that arise in many important applications such as modeling economies of scale in inventory management and supply chain management problems through concave or quasi-concave procurement costs. Moreover, several important optimization problems can be formulated as concave minimization problems including 0-1 integer programming [25], quadratic assignment [17] and [3], bilinear programming [13] and linear complementarity problem [18]. Therefore, concave or quasi-concave minimization forms an important class of non-convex optimization problems that are computationally intractable in general.

In this paper, we consider the problem of minimizing a *low-rank* quasi-concave function over a compact convex set. Let us first introduce a few definitions.

Definition 1.1 *A function $f : P \to \mathbb{R}$ over a convex set P is* quasi-concave *if and only if the upper level set*

$$U_\lambda = \{\boldsymbol{x} \in P \mid f(\boldsymbol{x}) \geq \lambda\},$$

is convex for all $\lambda \in \mathbb{R}$.

Note that any function that is concave is also quasi-concave; however, the converse is not true. For example, $f(x_1, x_2) = x_1 x_2$ is a quasi-concave function in $\mathbb{R}^2_+$ but is not concave [1]. Therefore, quasi-concave functions are a generalization of concave functions.

*These results were published as a Technical Report at Tepper Business School (Working Paper 366). V. Goyal is supported by NSF grant CMMI-120116 and R. Ravi is supported by NSF grants CCF-0728841and CCF-1143998.

Concave minimization is NP-hard even for the special case of minimizing a concave quadratic function over a hypercube [8]. In fact, Mittal and Schulz [20] show that it is not possible to approximate the minimum of a general concave function over the unit hypercube to within any factor, unless P = NP. Since the general concave minimization problem is hard, special cases of the problem have been of interest. To discuss the special cases, we first introduce the notion of a *rank* of a function following [15], and [24].

Definition 1.2 *The rank of a function $f : \mathbb{R}^n \to \mathbb{R}$ is the smallest integer k such that,*

$$f(\boldsymbol{x}) = g(\boldsymbol{a}_1^T\boldsymbol{x}, \boldsymbol{a}_2^T\boldsymbol{x}, \ldots, \boldsymbol{a}_k^T\boldsymbol{x}), \forall \boldsymbol{x} \in \mathbb{R}^n,$$

for some function $g : \mathbb{R}^k \to \mathbb{R}$ and linearly independent vectors $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_k \in \mathbb{R}^n$.

Below are some examples to illustrate the notion of the rank of a function.

1. If $f(\boldsymbol{x}) = (\boldsymbol{a}_1^T\boldsymbol{x})(\boldsymbol{a}_2^T\boldsymbol{x})$ where $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are linearly independent, then f has rank two.
2. If $f(\boldsymbol{x}) = \Pi_{i=1}^k(\boldsymbol{a}_i^T\boldsymbol{x})$ where $k \geq 1$ and $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k$ are linearly independent, then f has rank k.
3. If $f(\boldsymbol{x}) = \frac{\boldsymbol{a}_1^T\boldsymbol{x}}{\boldsymbol{a}_2^T\boldsymbol{x}}$ where $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are linearly independent and $\boldsymbol{a}_2^T\boldsymbol{x} \neq 0$, then f has rank two.

The problem of minimizing low-rank concave functions has been extensively studied in the literature. While one might think that the special case of constant-rank functions are easy to optimize, Pardalos and Vavasis [23] show that even minimizing a quadratic function of rank two over a polytope is NP-hard. Matsui [19] proves an even stronger version where he shows that minimizing the product of two strictly positive linear functions over a polytope is NP-hard. Therefore, it is natural to study algorithms that compute near-optimal solutions. This has received considerable interest in the literature both from a point of view of designing computationally efficient heuristics as well as designing polynomial time algorithms with provable performance bounds.

Konno et al. [14] consider the low-rank concave quadratic minimization problem and propose a tabu-search heuristic to solve the problem. Porembski [24] considers a generalization of the low-rank concave quadratic functions where the objective function is non-linear for a small number of variables and propose a cutting plane solution approach. Several other solution approaches including enumeration (see for instance [2] and [5]) and cutting plane methods [28] have been studied in the literature. We refer the reader to surveys by Horst and Pardalos [10] and Pardalos and Rosen [22] for an extensive discussion. The above solution approaches are efficient heuristics but do not provide any bounds on the running time or performance of the algorithm.

There has been extensive work from the perspective of designing efficient algorithms with provable performance bounds as well. Vavasis [29] gives an approximation scheme for low-rank quadratic optimization problems. However, the notion of approximation algorithm is different from the one we use in this paper. In [29], a solution $\hat{\boldsymbol{x}}$ is an ϵ-approximate solution for the problem of minimizing a function f if

$$f(\hat{\boldsymbol{x}}) \leq (1-\epsilon)f(\boldsymbol{x}^*) + \epsilon M,$$

where $M = \max_{\boldsymbol{x}} f(\boldsymbol{x})$. Therefore, any solution is 1-approximation according to this definition. On the other hand, we consider the standard notion of relative approximation ratio where a solution $\hat{\boldsymbol{x}}$ is a $(1+\epsilon)$-approximation if

$$f(\hat{\boldsymbol{x}}) \leq (1+\epsilon)f(\boldsymbol{x}^*).$$

Kern and Woeginger [12] consider the problem of minimizing a product of two linear functions and give an FPTAS for the problem. Recall that product of two linear functions is a quasi-concave function of rank 2.

Kelner and Nikolova [11] and Mittal and Schulz [20] are two recent papers that are most closely related to our work. Kelner and Nikolova [11] propose an approximation scheme with polynomial smoothed complexity for the low-rank quasi-concave minimization problem if the objective function satisfies a Lipschitz condition with respect to the L_1-norm with a polynomially bounded coefficient, i.e.,

$$|f(\boldsymbol{x}) - f(\boldsymbol{y}) \leq \phi(n) \cdot |\boldsymbol{x} - \boldsymbol{y}|_1, \ \forall \ \boldsymbol{x}, \boldsymbol{y},$$

where $\phi(\cdot)$ is a fixed polynomial and n is the dimension of the decision space. They also show that it is NP-hard to approximate the general quasi-concave minimization problem by a ratio better than $\Omega(\log n)$ unless P =NP. In our results, we do not require a Lipschitz condition on the objective but we do assume that the objective function satisfies certain scalability properties described later. The two assumptions are similar in spirit but are independent. It is possible that a function satisfies our assumption of scalability but not the Lipschitz condition of [11] and vice-versa. Moreover, our algorithm is a deterministic polynomial time algorithm and is based on solving several linear minimization problems with appropriate objectives that are based on the gradient of the original function. This approach provides geometric insights for the concave and quasi-concave minimization that are of independent interest and can be useful in designing practical algorithms for more general concave minimization problems.

In another recent paper, Mittal and Schulz [20] independently propose an FPTAS for optimizing a class of low-rank functions (not necessarily quasi-concave). Their algorithm is based on considering the projection in the low-rank subspace and solving feasibility problems at the set of appropriately chosen grid points.

The algorithm in [20] works for minimizing more general functions than just quasi-concave; however, it does not give an extreme point solution if the function is quasi-concave. To obtain an extreme point solution for quasi-concave functions, the authors refer to results of Diakonikolas and Yannakakis [6] who construct an approximate Pareto optimal curves efficiently in the context of multi-objective optimization. The problem of constructing an approximate Pareto optimal set for multi-objective optimization is also closely related and has been studied extensively (see Safer and Orlin [27], Papadimitriou and Yannakakis [21]). Moreover, the FPTAS presented in our paper is more efficient for quasi-concave functions under mild assumptions and we compare the running time in Section 2. Also, as mentioned earlier, our algorithm is based on solving a polynomial number of appropriately chosen linear optimization problems over a convex set and this technique could be of independent interest.

1.1 Model and Assumptions

We consider the problem of minimizing a quasi-concave function $f : \mathbb{R}^n \to \mathbb{R}_+$ of rank k over a compact convex set $P \subset \mathbb{R}^n$. We assume that the convex set P is specified by a separation oracle. Since f has rank k, there exists $g : \mathbb{R}^k \to \mathbb{R}_+$ and $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k \in \mathbb{R}^n$ such that

$$f(\boldsymbol{x}) = g(\boldsymbol{a}_1^T \boldsymbol{x}, \ldots, \boldsymbol{a}_k^T \boldsymbol{x}), \ \forall \boldsymbol{x} \in \mathbb{R}^n. \tag{1.1}$$

We assume that the linearly independent vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k$ that define the low-rank representation of f are given. Also, the function g and gradient ∇g are given as a black-box input. These can be readily computed for a large class of quasi-concave functions f. In addition, we assume that g satisfies the following conditions.

(P1) The gradient $\nabla g \geq 0$ for all $\boldsymbol{y} \in \mathbb{R}^k_+$, i.e. partial derivative

$$g_{y_i} = \frac{\partial g}{\partial y_i} \geq 0, \forall i = 1, \ldots, k, \ \forall \boldsymbol{y} \in \mathbb{R}^k_+$$

(P2) $g(\alpha \boldsymbol{y}) \leq \alpha^c g(\boldsymbol{y})$ for all $\alpha \geq 1$ and some constant c.

(P3) $\boldsymbol{a}_i^T \boldsymbol{x} > 0$ for all $i = 1, \ldots, k$, $\boldsymbol{x} \in P$.

The second assumption is similar in spirit to the Lipschitz condition assumed in [11]. The above conditions are satisfied by a large class of quasi-concave functions that include linear multiplicative functions [16] and utility functions from microeconomics theory such as Leontief utilities and Cobb-Douglas utilities [7]. We can now state our minimization problem, Π, as follows.

$$\min_{\boldsymbol{x} \in P} \; f(\boldsymbol{x}) = \min_{\boldsymbol{x} \in P} \; g(\boldsymbol{a}_1^T \boldsymbol{x}, \ldots, \boldsymbol{a}_k^T \boldsymbol{x}). \tag{1.2}$$

The following result about the minimum of quasi-concave functions is well known (for instance, see Bertsekas et al. [4]).

Proposition 1.3 *(Bertsekas et al. [4]) The minimum of a quasi-concave function over a compact convex set is attained at an extreme point of the set.*

In this paper, we present an FPTAS for the above problem where k is a constant and g satisfies (P1), (P2) and (P3) and relies on the above proposition. The basic idea of the algorithm is as follows. For a polynomial number of values of $\lambda \in \mathbb{R}$, we consider level curves, U_λ of g, where

$$U_\lambda = \{\boldsymbol{y} \mid g(\boldsymbol{y}) = \lambda\}.$$

For each level curve, we solve a polynomial number of linear optimization problems with appropriate objective functions that depend on the gradient of the function on the level curve. Using the properties of quasi-concavity and the fact that g satisfies (P1), (P2), and (P3), we prove that considering a polynomial number of different values of λ is sufficient to find a near-optimal solution to Π that is also an extreme point of P. Our results can also be extended to the problem of maximizing low-rank quasi-convex functions under similar assumptions.

Outline. In Section 2, we describe our FPTAS for minimizing constant rank quasi-concave functions. We discuss applications of our algorithm to combinatorial problems in Section 3.

2 $(1+\epsilon)$-Approximation Algorithm

We present a fully polynomial time approximation scheme (FPTAS) for the problem Π (1.2) in this section. Since f has rank k and can be expressed using $g : \mathbb{R}^k \to \mathbb{R}$ using an affine transformation through vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k$, we can reformulate Π as follows. Let

$$\boldsymbol{A} = [\boldsymbol{a}_1 \; \boldsymbol{a}_2 \cdots \boldsymbol{a}_k],$$

and

$$P_k = \{\boldsymbol{A}^T \boldsymbol{x} \mid \boldsymbol{x} \in P\}. \tag{2.1}$$

Since we have a separation oracle for P and $\boldsymbol{A}$ is known, we also have a separation oracle for P_k. Now, Π can be equivalently expressed as,

$$\begin{aligned} \min \quad & g(\boldsymbol{y}) \\ \text{s.t.} \quad & \boldsymbol{y} \in P_k. \end{aligned} \tag{2.2}$$

Note that $P_k \subset \mathbb{R}_+^k$ since $P \subset \mathbb{R}_+^n$ and $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k \in \mathbb{R}_+^n$. From any solution, $\boldsymbol{y} \in P_k$, we can construct a solution $\boldsymbol{x} \in P$ by solving a linear feasibility problem. We first show that g is also a quasi-concave function.

Lemma 2.1 *Let the function g be as defined in* (1.1)*. If the function f is quasi-concave, then g is also quasi-concave.*

Proof: Consider any $\lambda \in \mathbb{R}$ and consider the upper level set,

$$U_\lambda^g = \{\boldsymbol{y} \in P_k \mid g(\boldsymbol{y}) \geq \lambda\}.$$

Consider any $\boldsymbol{y}_1, \boldsymbol{y}_2 \in U_\lambda^g$ and $0 \leq \alpha \leq 1$ and let $\hat{\boldsymbol{y}} = \alpha \boldsymbol{y}_1 + (1-\alpha)\boldsymbol{y}_2$. We need to prove that $\hat{\boldsymbol{y}} \in P_k$. There exist $\boldsymbol{x}_1, \boldsymbol{x}_2 \in P$ such that $g(\boldsymbol{y}_j) = f(\boldsymbol{x}_j) \geq \lambda$ and $\boldsymbol{y}_j = \boldsymbol{A}^T \boldsymbol{x}_j$, $j = 1, 2$. Let

$$U_\lambda^f = \{\boldsymbol{x} \in P \mid f(\boldsymbol{x}) \geq \lambda\}.$$

Since f is quasi-concave, U_λ^f is convex. Therefore, $\hat{\boldsymbol{x}} = \alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2 \in P$ which implies $f(\hat{\boldsymbol{x}}) \geq \lambda$. Now,

$$\hat{\boldsymbol{y}} = \alpha \boldsymbol{y}_1 + (1-\alpha)\boldsymbol{y}_2 = \alpha \boldsymbol{A}^T \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{A}^T \boldsymbol{x}_2 = \boldsymbol{A}^T \hat{\boldsymbol{x}}.$$

Therefore, $\hat{\boldsymbol{y}} \in P_k$ which implies that g is quasi-concave. ■

Let us introduce a few notations. For all $i = 1, \ldots, k$,

$$\begin{aligned} \ell_i &= \min_{\boldsymbol{y} \in P_k} y_i \\ u_i &= \max_{\boldsymbol{y} \in P_k} y_i \\ \boldsymbol{y}_\ell^i &\leftarrow \mathsf{argmin}\{y_i \mid \boldsymbol{y} \in P_k\} \\ \boldsymbol{y}_u^i &\leftarrow \mathsf{argmax}\{y_i \mid \boldsymbol{y} \in P_k\}. \end{aligned} \tag{2.3}$$

Note that Assumption (P3) implies that $\ell_i > 0$ for all $i = 1, \ldots, k$. Let

$$H_k = [\ell_1, u_1] \times [\ell_2, u_2] \times \cdots \times [\ell_k, u_k].$$

and

$$H_{k-1} = [\ell_1, u_1] \times [\ell_2, u_2] \times \cdots \times [\ell_{k-1}, u_{k-1}].$$

Clearly, $P_k \subseteq H_k$. Note that $|\log \ell_i|$ and $|\log u_i|$ are both polynomial in the input size for all $i = 1, \ldots, k$. Therefore, we can construct a polynomial size grid to discretize the set H_{k-1}. In particular, for a given $\epsilon > 0$, let

$$n_i = \left\lceil \log_{(1+\epsilon)} \left(\frac{u_i}{\ell_i} \right) \right\rceil,$$

and

$$\Gamma_i^\epsilon = \{\ell_i \cdot (1+\epsilon)^j \mid j = 0, 1, \ldots, n_i\}. \tag{2.4}$$

Consider the following set $\Gamma^\epsilon \subset H_{k-1}$.

$$\Gamma^\epsilon = \Gamma_1^\epsilon \times \Gamma_2^\epsilon \times \ldots \times \Gamma_{k-1}^\epsilon. \tag{2.5}$$

Note that for any point $(y_1, \ldots, y_{k-1}) \in H_{k-1}$, there exists $(\tilde{y}_1, \ldots, \tilde{y}_{k-1}) \in \Gamma^\epsilon$ such that $y_j \leq (1+\epsilon)\tilde{y}_j$ for all $j = 1, \ldots, k-1$. Therefore, H_{k-1} can be *approximated* by the discrete set Γ^ϵ.

Now, we can describe the basic algorithm as follows. Let z^* denote the optimal objective value of (2.2). We maintain an upper bound, z_U and a lower bound, z_L on z^* and let

$$z = \frac{z_U + z_L}{2}.$$

In each iteration of the algorithm, we solve a polynomial number of linear optimization problems over P_k and either update the upper bound to less than or equal to z or update the lower bound to $z/(1 + O(\epsilon))$. In particular, we show that one of the following holds in each iteration of the algorithm.

1. Either the algorithm finds a feasible solution to Π with objective value less than or equal to z, or
2. $z^* \geq \frac{z}{1+O(\epsilon)}$.

Therefore, in each iteration we are able to reduce the ratio between the upper and lower bound by a constant factor and the algorithm terminates when z_U and z_L are close. So the number of iterations in the algorithm is $O\left(\log \frac{z_U}{z_L}\right)$. We give a formal description of the algorithm below. For any $\lambda \in \mathbb{R}$, let

$$\mathcal{T}_\lambda^\epsilon = \{(v_1, \ldots, v_k) \mid (v_1, \ldots, v_{k-1}) \in \Gamma^\epsilon, g(v_1, \ldots, v_k) = \lambda\}. \tag{2.6}$$

To compute $\mathcal{T}_\lambda^\epsilon$ for any $\lambda, \epsilon > 0$, we need to solve a single variable equation to compute v_k such that $g(v_1, \ldots, v_k) = \lambda$ given any $(v_1, \ldots, v_{k-1}) \in \Gamma^\epsilon$. We assume that this inverse function is given as a black-box input.

Input: A rank k quasi-concave function f, a compact convex set $P \subseteq \mathbb{R}^n_+$, and $\epsilon > 0$.

1. Let $\hat{\epsilon} = \epsilon/(c+1)$. Consider
$$\Gamma^{\hat{\epsilon}} = \Gamma_1^{\hat{\epsilon}} \times \Gamma_2^{\hat{\epsilon}} \times \ldots \times \Gamma_{k-1}^{\hat{\epsilon}}.$$
2. Initialize
$$z_L \leftarrow g(\ell_1, \ell_2, \ldots, \ell_k), \;\; z_U \leftarrow \min_{i=1}^{k} \{g(\boldsymbol{y}_\ell^i), g(\boldsymbol{y}_u^i)\}.$$
3. while $(z_U > (1+\epsilon) z_L)$
   (a) Let
   $$\lambda \leftarrow \frac{z_U + z_L}{2}.$$
   (b) For each $\boldsymbol{v} \in \mathcal{T}_\lambda^{\hat{\epsilon}}$,
      i. Solve the following linear minimization problem $\mathsf{LP}(\boldsymbol{v})$:
      $$\begin{aligned} z_{\boldsymbol{v}}^* = \min \; & (\nabla g(\boldsymbol{v}))^T \boldsymbol{y} \\ & \boldsymbol{y} \in P_k, \end{aligned}$$
      and let $\boldsymbol{y}_{\boldsymbol{v}}^* \in P_k$ be an extreme point optimal solution for $\mathsf{LP}(\boldsymbol{v})$.
      ii. If $z_U > g(\boldsymbol{y}_{\boldsymbol{v}}^*)$,
      $$z_U \leftarrow g(\boldsymbol{y}_{\boldsymbol{v}}^*), \;\; \boldsymbol{y}_{\mathcal{A}} \leftarrow \boldsymbol{y}_{\boldsymbol{v}}^*.$$
   (c) If for all $\boldsymbol{v} \in \mathcal{T}_\lambda^{\hat{\epsilon}}, z_{\boldsymbol{v}}^* \geq (\nabla g(\boldsymbol{v}))^T \boldsymbol{v}$, then
   $$z_L \leftarrow \frac{\lambda}{(1+\hat{\epsilon})^c}.$$
4. Return $\boldsymbol{y}_{\mathcal{A}}$.

Figure 1: **Algorithm $\mathcal{A}$ for Minimizing a low-rank Quasi-concave Function**

In the following lemmas, we prove the correctness of the algorithm. We first show that z_U and z_L are initialized correctly. Next, we show that the algorithm updates z_L and z_U correctly in each iteration. Finally, we show that in each iteration, the algorithm either updates z_U or z_L, and, therefore, reduces the ratio of z_U and z_L by a constant factor.

Lemma 2.2 *In Step 2 of Algorithm $\mathcal{A}$, the upper and lower bounds are initialized correctly, i.e.,*

$$g(\ell_1, \ell_2, \ldots, \ell_k) \leq z^* \leq \min_{i=1}^{k} \{g(\boldsymbol{y}_\ell^i), g(\boldsymbol{y}_u^i)\}.$$

Proof: Note that $\boldsymbol{y}_\ell^i, \boldsymbol{y}_u \in P_k$ for all $i = 1, \ldots, k$. Therefore, $z^* \leq \min\{g(\boldsymbol{y}_\ell^i), g(\boldsymbol{y}_u^i)\}$ for all i which implies

$$z^* \leq \min_{i=1}^{k} \{g(\boldsymbol{y}_\ell^i), g(\boldsymbol{y}_u^i)\}.$$

From (2.3), we know that for any $\boldsymbol{y} \in P_k$, $y_i \geq \ell_i$ for all $i = 1, \ldots, k$. Also, $\nabla g \geq 0$ for all $\boldsymbol{y} \in \mathbb{R}_+^k$. Therefore,

$$g(\ell_1, \ldots, \ell_k) \leq g(\boldsymbol{y}),\ \forall \boldsymbol{y} \in P_k,$$

which implies $g(\ell_1, \ldots, \ell_k) \leq z^*$ ■

Next, we show that z_U and z_L are correctly updated by the algorithm.

Lemma 2.3 *Algorithm $\mathcal{A}$ correctly updates z_U in Step 3(b)ii.*

Proof: Note that z_U is updated in Step 3(b)ii only if we find a feasible solution $\boldsymbol{y} \in P_k$ such that $z_U > g(\boldsymbol{y})$. Therefore, the algorithm updates z_U correctly ■

Lemma 2.4 *Algorithm $\mathcal{A}$ correctly updates z_L in Step 3c.*

Proof: In any iteration with parameter λ, z_L is updated only if for all $\boldsymbol{v} \in \mathcal{T}_\lambda^{\hat{\epsilon}}, z_{\boldsymbol{v}}^* \geq (\nabla g(\boldsymbol{v}))^T \boldsymbol{v}$. We show that in this case

$$\lambda \leq z^*(1+\hat{\epsilon})^c,$$

where z^* is the optimal value of (2.2). Suppose $\boldsymbol{y}^* \in P_k$ is an optimal solution of (2.2). By construction of $\Gamma^{\hat{\epsilon}}$, there exists $(v_1, \ldots, v_{k-1}) \in \Gamma^{\hat{\epsilon}}$ such that

$$\frac{v_i}{1+\hat{\epsilon}} \leq y_i^* < v_i,\ \forall i = 1, \ldots, k-1.$$

Let v_k be such that $g(v_1, \ldots, v_k) = \lambda$ and $\boldsymbol{v} = (v_1, \ldots, v_k) \in \mathcal{T}_\lambda^{\hat{\epsilon}}$. Recall that for all $i = 1, \ldots, k$, $g_{y_i}(\boldsymbol{v})$ denotes the partial derivative of g with respect to y_i at $\boldsymbol{v}$. Since $z_{\boldsymbol{v}}^* \geq (\nabla g(\boldsymbol{v}))^T \boldsymbol{v}$,

$$\begin{aligned}
\sum_{i=1}^{k} g_{y_i}(\boldsymbol{v}) v_i &\leq z_{\boldsymbol{v}}^* && (2.7)\\
&\leq \sum_{i=1}^{k} g_{y_i}(\boldsymbol{v}) y_i^* && (2.8)\\
&\leq \sum_{i=1}^{k-1} g_{y_i}(\boldsymbol{v}) v_i + g_{y_k}(\boldsymbol{v}) y_k^*, && (2.9)
\end{aligned}$$

where (2.8) follows from the optimality of $z_{\boldsymbol{v}}^*$ for $\mathsf{LP}(\boldsymbol{v})$ and (2.9) follows from the fact the $g_{y_i}(\boldsymbol{v}) \geq 0, y_i^* \leq v_i, \forall i = 1, \ldots, k-1$. Therefore, $v_k \leq y_k^*$ which implies $\boldsymbol{v} \leq (1+\hat{\epsilon})\boldsymbol{y^*}$. Now,

$$\lambda = g(\boldsymbol{v}) \leq g((1+\hat{\epsilon})\boldsymbol{y}^*) \leq (1+\hat{\epsilon})^c g(\boldsymbol{y}^*).$$

Here, the last inequality follows from the fact that $g(\alpha \boldsymbol{y}) \leq \alpha^c g(\boldsymbol{y}), \forall \alpha \geq 1$ and some constant c ■

Figure 2 illustrates the updates of lower and upper bounds in each iteration. Now, to complete the proof of the correctness, we show that in each iteration, we either update the upper bound or the lower bound.

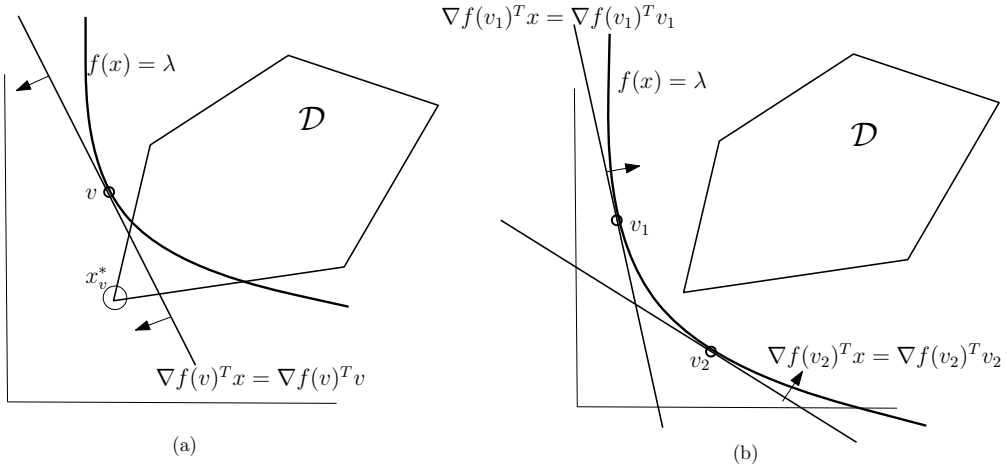

Figure 2: (a) $g(\boldsymbol{x}^*_{\boldsymbol{v}}) < \lambda \leq z_U$; (b) $z^*_{\boldsymbol{v}} \geq \nabla g(\boldsymbol{v})^T \boldsymbol{v}$ for all $\boldsymbol{v} \in \mathcal{T}_\lambda$

Lemma 2.5 *At the end of each iteration with $\lambda = (z_U + z_L)/2$ in the algorithm, one of the following holds.*

1. *Either z_U is updated to less than or equal to λ, or,*
2. *z_L is updated to $\lambda \cdot (1+\hat{\epsilon})^{-c}$ in step (3c).*

Proof: If $z^*_{\boldsymbol{v}} \geq (\nabla g(\boldsymbol{v}))^T \boldsymbol{v}$ for all $\boldsymbol{v} \in \mathcal{T}^{\hat{\epsilon}}_\lambda$, then from Lemma 2.4, we know that z_L is updated to $\lambda \cdot (1+\hat{\epsilon})^{-c}$. Suppose this is not the case. Then,

$$\exists\ \boldsymbol{v} \in \mathcal{T}^{\hat{\epsilon}}_\lambda \text{ s.t. } z^*_{\boldsymbol{v}} < (\nabla g(\boldsymbol{v}))^T \boldsymbol{v}.$$

We show that we can update the upper bound z_U in this case. Consider the upper level set

$$U_\lambda = \{\boldsymbol{y} \in \mathbb{R}^k_+ \mid g(\boldsymbol{y}) \geq \lambda\}.$$

Note that $\boldsymbol{v} \in U_\lambda$. Since g is quasi-concave, U_λ is convex. Therefore,

$$(\nabla g(\boldsymbol{v}))^T \boldsymbol{y} \geq (\nabla g(\boldsymbol{v}))^T \boldsymbol{v},$$

is a valid supporting hyperplane for U_λ [26]. In fact, it is a tangential hyperplane to the level curve $\{\boldsymbol{y} \in \mathbb{R}^k_+ \mid g(\boldsymbol{y}) = \lambda\}$ at $\boldsymbol{v}$. Since

$$z^*_{\boldsymbol{v}} = (\nabla g(\boldsymbol{v}))^T \boldsymbol{y}^*_{\boldsymbol{v}} < (\nabla g(\boldsymbol{v}))^T \boldsymbol{v},$$

$\boldsymbol{y}^*_{\boldsymbol{v}} \notin U_\lambda$ and $g(\boldsymbol{y}^*_{\boldsymbol{v}}) < \lambda$. Therefore, the upper bound z_U is updated to $g(\boldsymbol{y}^*_{\boldsymbol{v}})$ which is less than λ. ■

The following lemma bounds the running time of the algorithm.

Lemma 2.6 *The running time of the algorithm $\mathcal{A}$ is polynomial in input size of the problem and $\frac{1}{\epsilon}$.*

Proof: The algorithm does binary search for the objective value between the initial upper and lower bounds (z_U and z_L) until $z_U \leq z_L(1+\epsilon)$. In each iteration of the algorithm, the ratio of the upper and lower bounds reduces by at least $(1 - \hat{\epsilon}/2)$. If $z^0_U(z^0_L)$ denote the initial upper (lower) bound on the objective value, then the number of iterations is

$$O\left(\frac{1}{\hat{\epsilon}} \log \frac{z^0_U}{z^0_L}\right) = O\left(\frac{c}{\epsilon} \log \frac{z^0_U}{z^0_L}\right).$$

For each iteration with $\lambda = (z_L + z_U)/2$, we solve $O(|\mathcal{T}_\lambda^{\hat{\epsilon}}|)$ linear minimization problems over the convex set P. Let

$$R = \max_{i=1}^{k} \frac{u_i}{\ell_i}.$$

Note that $\log R$ is polynomial in the input size of the problem since both $|\log u_i|$ and $|\log \ell_i|$ are polynomial in input size for all $i = 1, \ldots, k$. Now,

$$|\mathcal{T}_\lambda^{\hat{\epsilon}}| \leq |\Gamma^{\hat{\epsilon}}| = \Pi_{i=1}^{k-1} |\Gamma_i^{\hat{\epsilon}}| = \left(\log_{1+\hat{\epsilon}} R\right)^{k-1} = O\left(\frac{\log R}{\hat{\epsilon}}\right)^{k-1} = O\left(\frac{c \log R}{\epsilon}\right)^{k-1}.$$

This shows that Algorithm $\mathcal{A}$ solves

$$O\left(\log \frac{z_U^0}{z_L^0} \cdot \frac{c^k \cdot (\log R)^{k-1}}{\epsilon^k}\right)$$

convex optimization problems (in fact linear optimization over a convex set) which is polynomial in the input size and $\frac{1}{\epsilon}$ for a constant k. ■

In contrast, the running time of the algorithm in [20] depends on $(\log R)^k$. We would like to note that we assume ∇g is given as a black-box input and the single-variable inverse function to compute $\mathcal{T}_\lambda^\epsilon$ is also given as a black-box input that are not required inputs for the algorithm in [20]. However, for a large class of functions f, ∇g and the inverse function are readily available.

Algorithm $\mathcal{A}$ returns $\boldsymbol{y}_\mathcal{A} \in P_k$ that is a $(1+\epsilon)$-approximation for minimizing g over P_k for any $\epsilon > 0$ where $\boldsymbol{y}_\mathcal{A}$ is an extreme point of P_k. To obtain a solution $\boldsymbol{x} \in P$ that is an extreme point near-optimal minimizer of f in P, we can solve the optimization problem over P.

$$\begin{aligned} \min\ & 0 \\ & \boldsymbol{A}^T \boldsymbol{x} = \boldsymbol{y}_\mathcal{A} \\ & \boldsymbol{x} \in P. \end{aligned}$$

Thus, we have the following theorem.

Theorem 2.7 *There is a fully polynomial time approximation scheme (FPTAS) for minimizing a quasi-concave function f of constant rank k over a convex set if f satisfies (P1), (P2) and (P3). Furthermore, the algorithm returns a solution that is an extreme point of the convex set. The running time is polynomial in the input size and $1/\epsilon$.*

3 Applications to 0-1 Combinatorial Problems

In this section, we consider the following combinatorial problem.

$$\min\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in S\}, \tag{3.1}$$

where $S \subseteq \{0,1\}^n$ and $f : \mathbb{R}_+^n \to \mathbb{R}_+$ is a quasi-concave function of rank k that satisfies assumptions (P1), (P2), and (P3). We further assume that we know the description of the convex hull of S and we can solve linear optimization problems over $\mathsf{conv}(S)$ in polynomial time. Our algorithm $\mathcal{A}$ directly applies to obtaining a fully polynomial time approximation scheme for (3.1) since $\mathcal{A}$ computes a solution that is an extreme point of the $\mathsf{conv}(S)$ which belongs to S.

Therefore, our result applies when S is the set of s-t paths, spanning trees, perfect matchings or s-t cuts in an undirected graph. Consider an undirected graph $G = (V, E)$, with costs $c_1 : E \to \mathbb{R}_+$, and $c_2 : E \to \mathbb{R}_+$. As an example, we obtain an FPTAS for the following problems.

1. **Spanning Trees**. Let $S \subset E$ be the set of spanning trees of G. Algorithm $\mathcal{A}$ can be adapted to obtain an FPTAS for
$$\min_{T \in S} c_1(T) \cdot c_2(T).$$

2. **Minimum Product Cut**. Let $S \subset E$ be the set of s-t cuts in G. Algorithm $\mathcal{A}$ can be adapted to obtain an FPTAS for
$$\min_{X \in S} c_1(X) \cdot c_2(X).$$

3. **Minimum Product Path**. Let $S \subset E$ be the set of s-t paths in G. Algorithm $\mathcal{A}$ can be adapted to obtain an FPTAS for
$$\min_{X \in S} c_1(X) \cdot c_2(X).$$

We would like to note that for the minimum product cut and minimum product path problems, we can optimize over the dominant of $\mathsf{conv}(S)$ instead of $\mathsf{conv}(S)$ since the costs are non-negative. The above problems can be generalized to minimizing a product of a constant number of linear costs instead of just two. This generalizes the result of Kern and Woeginger [12] and Goyal et al. [9] who consider minimizing the product of two linear functions over a polytope and give an FPTAS for the same. The general algorithm in [20] for low-rank minimization is not applicable for the above combinatorial problems as the solution is not necessarily an extreme point.

4 Conclusions

In this paper, we consider minimizing a class of low-rank quasi-concave functions over a convex set and obtain an FPTAS for the problem. Since f is minimized at an extreme point of the convex set, there exists a linear objective function that is minimized at the same extreme point as f. Our algorithm can be interpreted as an efficient search of such a linear objective function where we search over the gradient of f at a polynomial number of points. This search procedure is very general and it would be interesting to see if the idea can be extended to other non-convex optimization problems.

In comparison to [20], it gives better running times ($O((\log R)^{k-1})$ versus $O((\log R)^k)$ of [20]). Furthermore, it is also applicable to providing integer solutions of good quality to combinatorial problems with linear descriptions of their convex hulls as outlined in Section 3, whereas the general algorithm in [20] do not apply to this setting as their solution is not necessarily an extreme point.

References

[1] M. Avriel, WE Diewert, S. Schaible, and I. Zang. Generalized Convexity, 1988.

[2] M.L. Balinski. An algorithm for finding all vertices of convex polyhedral sets. *J. Soc. Indust. Appl. Math*, 9(1), 1961.

[3] M.S. Bazaraa and H.D. Sherali. On the use of exact and heuristic cutting plane methods for the quadratic assignment problem. *Journal of the Operational Research Society*, 33(1):991–1003, 1982.

[4] D.P. Bertsekas, A. Nedić, and A. Ozdaglar. Convex analysis and optimization. 2003.

[5] C.A. Burdet. Generating all the faces of a polyhedron. *SIAM Journal on Applied Mathematics*, 26(3):479–489, 1974.

[6] I. Diakonikolas and M. Yannakakis. Succinct approximate convex pareto curves. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 74–83. Society for Industrial and Applied Mathematics, 2008.

[7] E. Eisenberg. Aggregation of Utility Functions. *Management Sciences*, 7(4):337–350, 1961.

[8] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[9] V. Goyal, L. Genc-Kaya, and R. Ravi. An FPTAS for minimizing the product of two non-negative linear cost functions. *Mathematical Programming Ser. A*, pages 1–5, 2009.

[10] R. Horst and P.M. Pardalos. *Handbook of Global Optimization*. Kluwer Academic Publishers, 1995.

[11] Jonathan A. Kelner and Evdokia Nikolova. On the hardness and smoothed complexity of quasi-concave minimization. In *FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 472–482, Washington, DC, USA, 2007. IEEE Computer Society.

[12] W. Kern and G. Woeginger. Quadratic programming and combinatorial minimum weight product problems. *Mathematical Programming*, 110(3):641–649, 2007.

[13] H. Konno. A cutting plane algorithm for solving bilinear programs. *Mathematical Programming*, 11(1):14–27, 1976.

[14] H. Konno, C. Gao, and I. Saitoh. Cutting Plane/Tabu Search Algorithms for Low Rank Concave Quadratic Programming Problems. *Journal of Global Optimization*, 13(3):225–240, 1998.

[15] H. Konno, P.T. Thach, and T. Hoang. *Optimization on low rank nonconvex structures*. Kluwer Academic Boston, Mass, 1997.

[16] Hiroshi Konno and Takahito Kuno. Linear multiplicative programming. *Math. Program.*, 56(1-3):51–64, 1992.

[17] E.L. Lawler. The quadratic assignment problem. *Management Science*, 9(4):586–599, 1963.

[18] OL Mangasarian. The linear complementarity problem as a separable bilinear program. *Journal of Global Optimization*, 6(2):153–161, 1995.

[19] T. Matsui. NP-hardness of linear multiplicative programming and related problems. *Journal of Global Optimization*, 9(2):113–119, 1996.

[20] S. Mittal and A.S. Schulz. An FPTAS for Optimizing a Class of Low-Rank Functions Over a Polytope. *Mathematical Programming (Online First)*, 2012.

[21] C.H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 86–92. IEEE, 2000.

[22] P.M. Pardalos and JB Rosen. Methods for global concave minimization: A bibliographic survey. *SIAM Review*, 28(3):367–379, 1986.

[23] P.M. Pardalos and S.A. Vavasis. Quadratic programming with one negative eigenvalue is NP-hard. *Journal of Global Optimization*, 1(1):15–22, 1991.

[24] M. Porembski. Cutting Planes for Low-Rank-Like Concave Minimization Problems. *Operations Research*, 52(6):942–953, 2004.

[25] M. Raghavachari. On connections between zero-one integer programming and concave programming under linear constraints. *Operations Research*, 17(4):680–684, 1969.

[26] R.T. Rockefeller. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.

[27] H.M. Safer and J.B. Orlin. Fast approximation schemes for multi-criteria combinatorial optimization. *Technical Report*, 1995.

[28] H. Tuy. Concave programming under linear constraints. *Soviet Mathematics*, 5(1):1437–1440, 1964.

[29] S.A. Vavasis. Approximation algorithms for indefinite quadratic programming. *Mathematical Programming*, 57(1):279–311, 1992.