Implementation of an inference algorithm for second order models within OpenGM

Patrick Esser*
University of Heidelberg

October 29, 2012

Abstract

This document describes the implementation and evaluation of [KZ11] within the framework of OpenGM [ABK12].

Contents

1 Introduction

Many applications involve the optimization of a function g, i.e.

$$x^* \in \arg\max_x g(x)$$

Graphical Models are a common way to model the function g in order to exploit the structure of the problem. We restrict our attention to discrete additive models, which consist of a

*supervised by Joerg H. Kappes

bipartite graph $G = (V, F, E)$ and a finite label space $\mathcal{X}_v$ for each $v \in V$. They represent a function g as follows:

$$x = (x_v)_{v \in V} \in \mathcal{X} = \bigotimes_{v \in V} \mathcal{X}_v$$
$$g(x) = \sum_{f \in F} \theta_f \left((x_v)_{v \in N(f)} \right)$$

where $N(f)$ denotes the neighbours of factor f, i.e. $N(f) = \{v \in V \mid (v, f) \in E\}$. In the following we will assume $|N(f)| \leq 2$ for all $f \in F$ and $V = \{0, 1, \ldots, n-1\}$.

Various approaches to solve problem (1) exist. Quadratic programming relaxations [RL06] have the advantage of being exact and more compact compared to linear programming relaxations, but they are generally non-convex which makes global optimization hard. In [KZ11], the authors represent the quadratic program as a difference of convex functions to develop an algorithm for optimization. In the next section we will briefly discuss the most important properties of the algorithm.

2 The algorithm

2.1 QP relaxation

Since we are dealing with 2^{nd} order models only, we will identify factors with their corresponding neighbours, i.e. $f \equiv ij$ if $N(f) = \{i, j\}$. Let χ denote the characteristic function.

$$\chi(A)(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

Then we can write

$$\theta_{ij}(x_1, x_2) = \sum_{\substack{s_1 \in \mathcal{X}_i \\ s_2 \in \mathcal{X}_j}} \theta_{ij}(s_1, s_2) \chi(\{s_1\})(x_1) \chi(\{s_2\})(x_2)$$

for second order factors, and accordingly

$$\theta_i(x) = \sum_{s \in \mathcal{X}_i} \theta_i(s) \chi(\{s\})(x)$$

for first order factors. Introducing the set

$$\begin{aligned} C_I &= \left\{ p \in \mathbf{R}^m | \exists x \in \mathcal{X} \forall i \in V, s \in \mathcal{X}_i : p_{is} = \chi(\{s\})(x_i) \right\} \\ &= \left\{ p \in \mathbf{R}^m | \sum_{s \in \mathcal{X}_i} p_{is} = 1, p_{is} \in \{0, 1\} \right\} \end{aligned}$$

where $m = \sum_{v \in V} |\mathcal{X}_v|$, we get

$$\begin{aligned}
\max_{x \in \mathcal{X}} g(x) &= \max_{x \in \mathcal{X}} \left\{ \sum_{ij \in F} \theta_{ij}(x_i, x_j) + \sum_{i \in F} \theta_i(x_i) \right\} \\
&= \max_{p \in C_I} \left\{ \sum_{ij \in F} \sum_{\substack{s_1 \in \mathcal{X}_i \\ s_2 \in \mathcal{X}_j}} \theta_{ij}(s_1, s_2) p_{is_1} p_{js_2} + \sum_{i \in F} \sum_{s \in \mathcal{X}_i} \theta_i(s) p_{is} \right\} \\
&= \max_{p \in C_I} \{ \langle p, \Theta p \rangle + \langle \theta, p \rangle \}
\end{aligned}$$

for an appropriate matrix Θ, resp. an appropriate vector θ containing the function values.

Using the obvious bijection between C_I and $\mathcal{X}$, problem (1) is equivalent to

$$p^* \in \arg\max_{p \in C_I} \{ \langle p, \Theta p \rangle + \langle \theta, p \rangle \}$$

We will use the letter h to denote the function $\mathbf{R}^m \to \mathbf{R}, p \mapsto \{ \langle p, \Theta p \rangle + \langle \theta, p \rangle \}$.

Relaxing our set C_I, we arrive at the quadratic programming formulation:

$$p^* \in \arg\max_{p \in C} h(p) \tag{1}$$

$$C = \left\{ p \in \mathbf{R}^m | \sum_{s \in \mathcal{X}_i} p_{is} = 1, p_{is} \geq 0 \right\} \tag{2}$$

This formulation has a natural interpretation in terms of probability. For any $p \in C$, let $(X_i^p)_{i \in V} : (\Omega, P) \to \bigotimes_{i \in V} \mathcal{X}_i$ be independent random variables such that $P_{X_i^p}(A) = \sum_{s \in A} p_{is}$ for $A \in 2^{\mathcal{X}_i}$. By linearity we obtain that the expectation of $g(X^p)$ is given by h:

$$\begin{aligned}
\mathbb{E}(g(X^p)) &= \sum_{ij \in F} \mathbb{E}_p(\theta_{ij}(X^p)) + \sum_{i \in F} \mathbb{E}_p(\theta_i(X^p)) \\
&= \sum_{ij \in F} \sum_{\substack{s_1 \in \mathcal{X}_i \\ s_2 \in \mathcal{X}_j}} \theta_{ij}(s_1, s_2) P_{X_i^p}(s_1) P_{X_j^p}(s_2) + \sum_{i \in F} \sum_{s \in \mathcal{X}_i} \theta_i(s) P_{X_i^p}(s) \\
&= h(p)
\end{aligned}$$

Following [RL06] we will briefly proof the most important properties of this relaxation. First of all, we will show that we can efficiently recover an integer solution (i.e. a $p \in C_I$) out of a relaxed solution, without decreasing the objective.

Theorem 2.1. *Define p^n recursively by*

$$p_{is}^{t+1} = \begin{cases} p_{is}^t & \text{for } i \neq t \\ \chi\left(\arg\max_{s' \in \mathcal{X}_t} \left\{ \sum_{j \in N(t)} \sum_{s_2 \in \mathcal{X}_j} \theta_{tj}(s', s_2) p_{js_2}^t + \theta_t(s') \right\} \right)(s) & \text{for } i = t \end{cases}$$

Then for any starting value $p^0 \in C$ the following holds true:

1. $p^n \in C_I$
2. $h(p^0) \leq h(p^n)$

Proof.

1. By definition of p^{t+1} it is clear that $p^{t+1}_{is} \in \{0,1\}$ for all $0 \leq i < t, s \in \mathcal{X}_i$ and $p^{t+1} \in C$ if $p^t \in C$. This immediately gives the result.

2. We will show $h(p^t) \leq h(p^{t+1})$.

   Let $x_t := \arg\max_{s' \in \mathcal{X}_t} \left\{ \sum_{j \in N(t)} \sum_{s_2 \in \mathcal{X}_j} \theta_{tj}(s', s_2) p^t_{js_2} + \theta_t(s') \right\}$, then

$$\begin{aligned}
h(p^{t+1}) &= \sum_{ij \in F} \sum_{\substack{s_1 \in \mathcal{X}_i \\ s_2 \in \mathcal{X}_j}} \theta_{ij}(s_1, s_2) p^{t+1}_{is_1} p^{t+1}_{js_2} + \sum_{i \in F} \sum_{s \in \mathcal{X}_i} \theta_i(s) p^{t+1}_{is} \\
&= \sum_{\substack{ij \in F \\ i,j \neq t}} \sum_{\substack{s_1 \in \mathcal{X}_i \\ s_2 \in \mathcal{X}_j}} \theta_{ij}(s_1, s_2) p^t_{is_1} p^t_{js_2} + \sum_{i \in F \setminus \{t\}} \sum_{s \in \mathcal{X}_i} \theta_i(s) p^t_{is} + \\
&+ \sum_{j \in N(t)} \sum_{s \in \mathcal{X}_j} \theta_{tj}(x_t, s) p^t_{js} + \theta_t(x_t) \\
&= h(p^t) + \sum_{j \in N(t)} \sum_{s \in \mathcal{X}_j} \theta_{tj}(x_t, s) p^t_{js} + \theta_t(x_t) + \\
&- \left(\sum_{j \in N(t)} \sum_{\substack{s_1 \in \mathcal{X}_t \\ s_2 \in \mathcal{X}_j}} \theta_{tj}(s_1, s_2) p^t_{ts_1} p^t_{js_2} + \sum_{s \in \mathcal{X}_t} \theta_t(s) p^t_{ts} \right)
\end{aligned}$$

   Looking at the last term we obtain

$$\begin{aligned}
&\sum_{j \in N(t)} \sum_{\substack{s_1 \in \mathcal{X}_t \\ s_2 \in \mathcal{X}_j}} \theta_{tj}(s_1, s_2) p^t_{ts_1} p^t_{js_2} + \sum_{s \in \mathcal{X}_t} \theta_t(s) p^t_{ts} \\
&= \sum_{s_1 \in \mathcal{X}_t} p^t_{ts_1} \left(\sum_{j \in N(t)} \sum_{s_2 \in \mathcal{X}_j} \theta_{tj}(s_1, s_2) p^t_{js_2} + \theta_t(s_1) \right) \\
&\leq \max_{s \in \mathcal{X}_t} \left\{ \sum_{j \in N(t)} \sum_{s_2 \in \mathcal{X}_j} \theta_{tj}(s, s_2) p^t_{js_2} + \theta_t(s) \right\} \underbrace{\sum_{s_1 \in \mathcal{X}_t} p^t_{ts_1}}_{=1} \\
&= \sum_{j \in N(t)} \sum_{s_2 \in \mathcal{X}_j} \theta_{tj}(x_t, s_2) p^t_{js_2} + \theta_t(x_t)
\end{aligned}$$

   Substituting this inequality into the previous equation finishes the proof.

□

Corollary 2.2. *The quadratic programming formulation is exact, i.e.*

$$\max_{p \in C_I} h(p) = \max_{p \in C} h(p)$$

Proof. Since $C_I \subset C$, it is clear that $\max_{p \in C_I} h(p) \leq \max_{p \in C} h(p)$. The other inequality follows from Theorem 2.1. □

In general, the matrix Θ in (1) is not negative definite and therefore we have to maximize a non-concave function. While [RL06] introduce a concave approximation to deal with this problem, [KZ11] derive an algorithm based on *DCA* which can work with the non-concave as well as with the concave version.

2.2 DCA/CCCP

DCA (difference of convex functions algorithm) is an algorithm for minimizing a function f that can be written as $f = u - v$, where u and v are lower-semicontinuous, proper, convex functions [AT05]. The special case of v being differentiable leads to a simpler algorithm, which is also called *CCCP* (convex-concave procedure) [SL09]. We will state its most important properties for our case without proofs. For proofs, see [SL09] or [AT05] in the more general setting.

Theorem 2.3. *Let $f = u - v$, with u, v strictly convex and continuously differentiable functions, $C = \{x \in \mathbf{R}^n | Ax = b, Cx \leq d\}$ s.t. C is bounded. Define $(x^i)_{i \in N}$ by*

$$x^{i+1} \in \arg\min_{x \in C} \{u(x) - \langle Dv(x^i), x \rangle\}$$

Then for any $x^0 \in C$, the following holds true

1. $f(x^{i+1}) \leq f(x^i)$
2. $f(x^i) \to f(x^*)$ *for some stationary point* x^*
3. $|x^i - x^{i+1}| \to 0$

As the existence of a decomposition into the difference of convex functions implies that there are infinitely many such decompositions, choosing one remains an art. Furthermore the algorithm does not give any lower bounds on f, so nothing is known about the quality of the solution.

The decomposition of [KZ11] requires the second order factors in (1) to be positive. Since adding a constant does not change the point where the maximum is attained we can assume $\theta_{ij} > 0$ by subtracting $min\{\theta_f((s_i)) | f \in F, (s_i) \in \bigotimes_{i \in N(f)} \mathcal{X}_i\} - \epsilon$ from each factor if necessary. Introducing the functions

$$u : p \mapsto \sum_{ij} \sum_{s_1, s_2} \frac{1}{2} \theta_{ij}(s_1, s_2)(p_{is_1}^2 + p_{js_2}^2) + \sum_i \sum_s \frac{1}{2} \theta_i(s)(p_{is}^2 + 1)$$

$$v : p \mapsto \sum_{ij} \sum_{s_1, s_2} \frac{1}{2} \theta_{ij}(s_1, s_2)(p_{is_1} + p_{js_2})^2 + \sum_i \sum_s \frac{1}{2} \theta_i(s)(p_{is} + 1)^2$$

we see that

$$u(p) - v(p) = -h(p)$$

and since we assumed $\theta_{ij} > 0$, u and v are strictly convex. Hence (1) is equivalent to

$$p^* \in \arg\min_{p \in C} \{u(p) - v(p)\} \tag{3}$$

As suggested by Theorem 2.3, we want to solve

$$p^{i+1} \in \arg\min_{p \in C} \{u(p) - \langle Dv(p^i), p \rangle\} \tag{4}$$

Since this is a convex program with linear equality and inequality constraints, the *KKT* (Karush-Kuhn-Tucker) conditions are both necessary and sufficient (see any text on optimization). In particular, the inequality constraints introduce nonlinear conditions, known as the *complementary slackness conditions*. Those can be hard to solve for and the strength of the chosen decomposition lies in the possibility of iteratively optimizing with equality constraints only and still obtaining a solution of (4).

Consider the sequence (p^k) defined by

$$p^{k+1} \in \arg\min_{p \in C^k} \{u(p) - \langle Dv(\hat{p}), p \rangle\} \tag{5}$$

$$C^{k+1} = C^k \cap \{\ p \mid p_{is} = 0 \text{ for all } is \text{ s.t. } p_{is}^{k+1} < 0\ \} \tag{6}$$

with $p^0 \in C$ arbitrary and $C^0 = \{\ p \mid \sum_s p_{is} = 1\ \}$ is our original set C without the inequality constraints. We will show that this iteration scheme can be used to obtain a solution for (4). Problem (5) gives rise to the following Lagrangian:

$$L^{k+1}(p, \lambda, \bar{\lambda}) = u(p) - \langle Dv(\hat{p}), p \rangle + \langle \lambda, (\sum_s p_{is} - 1)_i \rangle + \langle \bar{\lambda}, (p_{is})_{is \in I^k} \rangle$$

where I^k denotes the set of all indices is for which p_{is} is constrained to be zero by C^k, i.e. $I^k = \left\{ is | \exists 1 \leq l \leq k : p_{is}^l < 0 \right\}$. Using the first order optimality conditions for convex programs with equality constraints, we obtain the following characterisation of p^{k+1}:

$$p^{k+1} \in \arg\min_{p \in C^k} \{u(p) - \langle Dv(\hat{p}), p \rangle\} \tag{7}$$

$$\iff \begin{cases} D_{is}u(p^{k+1}) - D_{is}v(\hat{p}) + \lambda_i^{k+1} + \chi(I^k)(is)\bar{\lambda}_{is} & = 0 \text{ for some } \lambda^{k+1},\ \bar{\lambda}^{k+1} \\ \sum_s p_{is}^{k+1} & = 1 \\ p_{is}^{k+1} & = 0 \text{ for all } is \in I^k \end{cases} \tag{8}$$

$$\iff \begin{cases} p_{is}^{k+1} & = \chi\left((I^k)^c\right)(is)\left(\frac{D_{is}v(\hat{p}) - \lambda_i^{k+1}}{\overline{\theta_{is}}}\right) \\ \lambda_i^{k+1} & = \left(\sum_{s \in (I_i^k)^c} \frac{1}{\overline{\theta_{is}}}\right)^{-1}\left(\sum_{s \in (I_i^k)^c} \frac{D_{is}v(\hat{p})}{\overline{\theta_{is}}} - 1\right) \end{cases} \tag{9}$$

where $\overline{\theta_{is}} = \sum_{j \in N(i)} \sum_{s_2 \in \mathcal{X}_j} \theta_{ij}(s, s_2) + \theta_i(s) > 0$ and $I_i^k = \{s \in \mathcal{X}_i | is \in I^k\}$. We have the following result:

Theorem 2.4. *Let (p^k) be given by (5) with $p^0 \in C$. For $k_0 = \min\{k \geq 1 | p^k \in C\}$ we have*

$$p^{k_0} \in \arg\min_{p \in C} \{u(p) - \langle Dv(\hat{p}), p \rangle\}$$

Furthermore, k_0 is bounded by m.

Proof. The last statement is clear, since p^k satisfies $\sum_s p_{is}^k = 1$ by definition, and it has at least $k-1$ entries set to zero or one.

To show that p^{k_0} is optimal, we will obtain multipliers λ_i, μ_{is}, satisfying the remaining *KKT* conditions:

$$D_{is}u(p^{k_0}) - D_{is}v(\hat{p}) + \lambda_i - \mu_{is} = 0 \qquad \text{for all } is \tag{10}$$

$$\mu_{is} p_{is}^{k_0} = 0 \qquad \text{for all } is \tag{11}$$

$$\mu_{is} \geq 0 \qquad \text{for all } is \tag{12}$$

Choosing $\lambda_i^{k_0}$ as in (9) we obtain from (8):

$$D_{is}u(p^{k_0}) - D_{is}v(\hat{p}) + \lambda_i^{k_0} = 0 \qquad \text{for } is \notin I^{k_0-1}$$

So we can use $\mu_{is} := 0$ for $is \notin I^{k_0-1}$. For $is \in I^{k_0-1}$ we define

$$\mu_{is} := D_{is}u(p^{k_0}) - D_{is}v(\hat{p}) + \lambda_i^{k_0} \qquad \text{for } is \in I^{k_0-1}$$

This ensures (10). Since $p_{is}^{k_0} = 0$ for $is \in I^{k_0-1}$, (11) holds as well, so all that remains to be shown is (12).

Since $is \in I^{k_0-1}$, there must be $1 \leq l \leq k_0 - 1$ such that $p_{is}^{l-1} \geq 0$ and $p_{is}^l < 0$. That means $is \notin I^{l-1}$ and therefore

$$\begin{aligned}
0 &= D_{is}u(p^l) - D_{is}v(\hat{p}) + \lambda_i^l \\
&= \underbrace{p_{is}^l}_{<0} \underbrace{\overline{\theta_{is}}}_{\geq 0} - D_{is}v(\hat{p}) + \lambda_i^l \\
&\leq -D_{is}v(\hat{p}) + \lambda_i^l \\
&= \underbrace{p_{is}^{k_0}\overline{\theta_{is}}}_{=0} - D_{is}v(\hat{p}) + \underbrace{\lambda_i^{k_0} - \lambda_i^{k_0}}_{=0} + \lambda_i^l \\
&= \mu_{is} + \lambda_i^l - \lambda_i^{k_0}
\end{aligned}$$

From this, we see that the result follows if $\lambda_i^l \leq \lambda_i^{k_0}$ is true for $1 \leq l \leq k_0 - 1$. The monotonicity of the multipliers λ_i can be seen as folllows:

$$\begin{aligned}
1 = \sum_s p_{is}^k &= \sum_{s \in I_i^k} \underbrace{p_{is}^k}_{<0} + \sum_{s \notin I_i^k} p_{is}^k && \leq \sum_{s \notin I_i^k} p_{is}^k \\
1 = \sum_s p_{is}^{k+1} &= \sum_{s \in I_i^k} \underbrace{p_{is}^{k+1}}_{=0} + \sum_{s \notin I_i^k} p_{is}^{k+1} && = \sum_{s \notin I_i^k} p_{is}^{k+1} \\
\Rightarrow \qquad 0 \leq \sum_{s \notin I_i^k} \left(p_{is}^k - p_{is}^{k+1}\right) &= \sum_{s \notin I_i^k} \frac{\lambda_i^{k+1} - \lambda_i^k}{\overline{\theta_{is}}} && = \left(\lambda_i^{k+1} - \lambda_i^k\right) \sum_{s \notin I_i^k} \frac{1}{\overline{\theta_{is}}}
\end{aligned}$$

For any $k \geq 1$, which implies the desired result. □

Collecting all equations and eliminating unnecessary iteration indices, we arrive at the final algorithm:

Algorithm 1 the complete algorithm

for all $i \in V$ **do**
  for all $s \in \mathcal{X}_i$ **do**
    $\overline{\theta_{is}} := \sum_{j \in N(i)} \sum_{s' \in \mathcal{X}_j} \theta_{ij}(s, s') + \theta_i(s)$
repeat
  for all $i \in V$ **do**
    for all $s \in \mathcal{X}_i$ **do**
      $D_{is}v := p_{is}\overline{\theta_{is}} + \sum_{j \in N(i)} \sum_{s' \in \mathcal{X}_j} \theta_{ij}(s, s')p_{js'} + \theta_i(s)$
  for all $i \in V$ **do**
    $I := \emptyset$
    repeat
      $\lambda_i := \left(\sum_{s \in \mathcal{X}_i \setminus I} \frac{1}{\overline{\theta_{is}}}\right)^{-1} \left(\sum_{s \in \mathcal{X}_i \setminus I} \frac{D_{is}v}{\overline{\theta_{is}}} - 1\right)$
      for all $s \in \mathcal{X}_i$ **do**
        if $s \in I$ **then**
          $p_{is} := 0$
        else
          $p_{is} := \frac{D_{is}v - \lambda_i}{\overline{\theta_{is}}}$
          if $p_{is} < 0$ **then**
            $I := I \cup \{s\}$
    until $0 \leq p_{is} \leq 1$ for all $s \in \mathcal{X}_i$ and $\sum_{s \in \mathcal{X}_i} p_{is} = 1$
until convergence

3 Implementation

3.1 OpenGM

OpenGM [ABK12] is a C++ library for graphical models. While the implemented algorithm, which we call *QPDC* (quadratic programming with difference of convex functions), is designed to work only on 2^{nd} order models that factorize additive, OpenGM can store graphical models of arbitrary order and size. Providing a uniform interface to graphical models, algorithms can be implemented independently of the specific model, making code reusable. Furthermore, OpenGM can store its models as *hdf5* files and includes a command line interface, which provides a good workflow for comparing different algorithms.

To meet the requirement of positive factors, we introduced a wrapper around the factor evaluation provided by OpenGM, which also provides a convenient way to switch between minimization and maximization. We provided options to set the maximum number of iterations as well as a threshold for considering the solution as converged (based on the squared $l2$-metric of two consecutive solutions, which is guaranteed to converge by Theorem 2.3). The convex approximation as described in [RL06] can be used and various methods for choosing a starting point are provided.

3.2 Modifications

After running algorithm 1, we noticed that the biggest improvement of the objective occurs when rounding the solution according to Theorem 2.1. This caused us to to introduce more options.

One option enables the rounding of the starting value until the objective does not improve anymore by the rounding process. This closes the big gap between the expectation and the corresponding value that was observed in most datasets (see figure 1). While this method obviously converges (since there are only finitely many values for the objective), we do not know any theoretical results about its efficiency, which made us introduce an option to limit the number of roundings done before starting the algorithm.

The last option we introduced closes any gap between the expectation and the corresponding value of the objective that develops during the algorithm. These modifications do not affect the algorithm's ability to find a stationary point, since we round the current solution only if there is a gap, i.e. if it strictly improves our objective. But again, this can only happen finitely many times.

4 Evaluation

To compare our results to the ones reported in [KZ11], we converted the *Rosetta protein design* and *side chain prediction* (SCP) datasets of [YMW06] into OpenGM models. The authors of [KZ11] report the solution quality of their algorithm based on the percentage of the optimal solution. Unfortunately we could not find the optimal values for the design dataset (except for two instances) and therefore we could not compare our solutions to theirs. Furthermore we note that this metric does not seem to be appropriate, since adding a constant to the modelled function results in all solutions obtaining a better quality in this metric.

In its unmodified form, the algorithm did not converge within 1500 iterations on any of the protein design instances, regardless of the starting point. This is in contrast to the findings of [KZ11], who report convergence within 1200 iterations on all of the instances.

Using the modifications described in section 3.2, the algorithm converged on all instances within 10 iterations and found better optima than the one obtained by the unmodified version after 1500 iterations. The mean running time for the modified version was 13.4 seconds (with a maximum running time of 32.3 seconds), where the unmodified version had a mean running time of 1513 seconds after 1500 iterations.

Looking at a typical run of the unmodified version (see figure 1), we see that it exhibits sublinear convergence. In fact, running the algorithm for 70000 iterations on an instance of the protein design dataset, which took 21 hours, did not result in convergence. This is not because of the choice of convergence threshold, because the integer solution and its corresponding value still change after 70000 iterations.

The results on the SCP dataset are similiar. Since the modified version converges very fast, we can run the algorithm with different starting points to obtain good results. See figure 2, where we used $\frac{f(x^*)-f_{min}}{f_{max}-f_{min}}$ as a measurement of the quality for solution x^*. Since the global maximum was unknown to us we used the maximal value found while running the

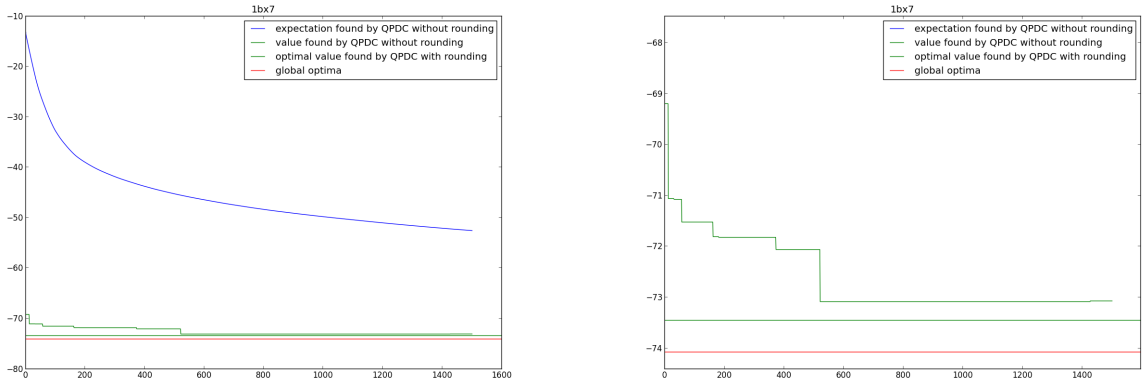

Figure 1: Typical run of the unmodified algorithm on the Rosetta Protein Design dataset.

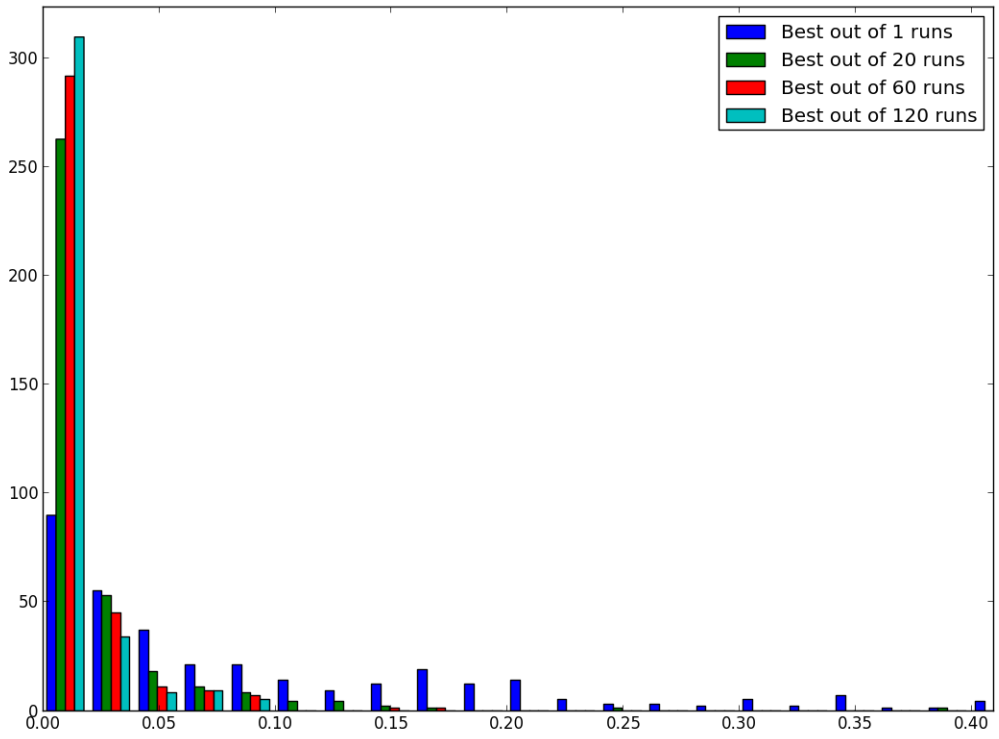

Figure 2: Distribution of solution quality found by the modified algorithm for the SCP dataset. Running the algorithm 120 times required 26.4 seconds on average.

algorithm to approximate the denominator.

Our results with the convex approximation coincide with [KZ11]. The convex approximation produced decent results on the design dataset and bad results on the SCP dataset. It converged in both cases within 40 iterations.

Although it seems like the main work is being done by the rounding process instead of CCCP, the rounding on its own does not necessarily find a stationary point. But in conjunction with CCCP this provides a very fast algorithm, which could be used as the basis for a global optimization method.

References

[ABK12] Bjoern Andres, Thorsten Beier, and Joerg H. Kappes, *Opengm 2.0 manual*, 2012.

[AT05] Le An and Pham Tao, *The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems*, Annals of Operations Research **133** (2005), 23–46, 10.1007/s10479-004-5022-1.

[KZ11] Akshat Kumar and Shlomo Zilberstein, *Message-passing algorithms for quadratic programming formulations of MAP estimation*, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (Barcelona, Spain), 2011, pp. 428–435.

[RL06] Pradeeep Ravikumar and John Lafferty, *Quadratic programming relaxations for metric labeling and markov random field map estimation*, Machine Learning: Proceedings of the Twenty-Third International Conference (ICML), 2006.

[SL09] Bharath K. Sriperumbudur and Gert R. G. Lanckriet, *On the convergence of concave-convex procedure*, In NIPS Workshop on Optimization for Machine Learning, 2009.

[YMW06] Chen Yanover, Talya Meltzer, and Yair Weiss, *Linear programming relaxations and belief propagation – an empirical study*, Journal of Machine Learning Research (2006).