Submodular semilinear valued constraint satisfaction problems

Caterina Viola

TU-Dresden
Institut für Algebra

November 2016

VCSPs

Let D be a fixed set (called domain or set of labels).
A cost function over D is any $f : D^n \to \mathbb{Q}$, for any $n \in \mathbb{N}$.

VCSPs

Let D be a fixed set (called domain or set of labels).
A cost function over D is any $f : D^n \to \mathbb{Q}$, for any $n \in \mathbb{N}$.
(Valued constraint) language: a finite set Γ of cost functions over D.

VCSPs

Let D be a fixed set (called domain or set of labels).
A cost function over D is any $f : D^n \to \mathbb{Q}$, for any $n \in \mathbb{N}$.
(Valued constraint) language: a finite set Γ of cost functions over D.

The valued constraint satisfaction problem for Γ, VCSP(Γ), is a computational optimisation problem.
INPUT:

- a finite set $V = \{x_1, \dots, x_n\}$ of variables, and
- an objective function $\Phi(x_1, \dots, x_n) = \sum_{i=1}^{k} f_i(x_{i,1}, \dots, x_{i,q_i})$, where $1 \le i \le k$, $x_{i,j} \in V$ and f_i is a cost function over D.

GOAL: find an assignment of labels (or labeling) to the variables that minimises Φ.

What we know

Let Γ be a valued constraint language over a finite domain D.
The computational complexity of VCSP(Γ) has attracted a lot of attention in the literature. All partial classifications were subsumed and generalised by:

Theorem (Thapper and Živný, 2013)

VCSP(Γ) *is either polynomial-time solvable or NP-hard.*

What we know

Let Γ be a valued constraint language over a finite domain D.
The computational complexity of VCSP(Γ) has attracted a lot of attention in the literature. All partial classifications were subsumed and generalised by:

Theorem (Thapper and Živný, 2013)

VCSP(Γ) is either polynomial-time solvable or NP-hard.

What happens in infinite domains?

What we know

Let Γ be a valued constraint language over a finite domain D.
The computational complexity of VCSP(Γ) has attracted a lot of attention in the literature. All partial classifications were subsumed and generalised by:

Theorem (Thapper and Živný, 2013)

VCSP(Γ) is either polynomial-time solvable or NP-hard.

What happens in infinite domains?

Our goal is classify the computational complexity of VCSPs for semilinear languages.

Semilinear VCSPs

A function $f : \mathbb{Q}^n \to \mathbb{Q}$ is semilinear if it is first-order definable over $(\mathbb{Q}; \leq, +, 1)$.

Semilinear VCSPs

A function $f : \mathbb{Q}^n \to \mathbb{Q}$ is semilinear if it is first-order definable over $(\mathbb{Q}; \leq, +, 1)$.

Example

$f : \mathbb{Q}^3 \to \mathbb{Q}$

$$f(x,y,z) = \begin{cases} 5x + 7z & \text{if} \quad x + y \leq 3 \\ -2 & \text{if} \quad x + y > 3 \text{ and } \max(x,y) > z + 1 \\ \min(2y + 6, -1) & \text{otherwise} \end{cases}$$

Semilinear VCSPs

A function $f\colon \mathbb{Q}^n \to \mathbb{Q}$ is semilinear if it is first-order definable over $(\mathbb{Q}; \leq, +, 1)$.

Example

$f\colon \mathbb{Q}^3 \to \mathbb{Q}$

$$f(x,y,z) = \begin{cases} 5x + 7z & \text{if} \quad x + y \leq 3 \\ -2 & \text{if} \quad x + y > 3 \text{ and } \max(x,y) > z + 1 \\ \min(2y + 6, -1) & \text{otherwise} \end{cases}$$

In a semilinear VCSP the underlying domain is $\mathbb{Q}$ and the language is made up by semilinear cost functions.

The Thapper&Živný's dichotomy

Let D be a finite set.

- Either Γ has a **symmetric fractional polymorphism** and VCSP(Γ) is in P,
- or VCSP(Γ) is NP-hard.

Fractional polymorphisms

Let $\mathcal{O}_D^{(m)}$ denote the set of all m-ary operations $g\colon D^m \to D$, for $m \in \mathbb{N}$.

Fractional polymorphisms

Let $\mathcal{O}_D^{(m)}$ denote the set of all m-ary operations $g\colon D^m \to D$, for $m \in \mathbb{N}$.

An m-ary fractional operation ω on D is a probability distribution on $\mathcal{O}_D^{(m)}$.
The support of ω is defined as $Supp(\omega) = \{g \in \mathcal{O}_D^{(m)} \mid \omega(g) > 0\}$.

Fractional polymorphisms

Let $\mathcal{O}_D^{(m)}$ denote the set of all m-ary operations $g\colon D^m \to D$, for $m \in \mathbb{N}$.

An m-ary fractional operation ω on D is a probability distribution on $\mathcal{O}_D^{(m)}$. The support of ω is defined as $Supp(\omega) = \{g \in \mathcal{O}_D^{(m)} \mid \omega(g) > 0\}$.

A m-ary fractional operation ω on D with finite support is said to be a fractional polymorphism of a cost function f if, for any $x_1, x_2, \ldots, x_m \in D^n$, we have

$$\sum_{g \in Supp(\omega)} \omega(g) f(g(x_1, x_2, \ldots, x_m)) \leq \frac{1}{m}(f(x_1) + f(x_2) + \cdots + f(x_m)),$$

where g is applied componentwise.

Fractional polymorphisms

Let $\mathcal{O}_D^{(m)}$ denote the set of all m-ary operations $g \colon D^m \to D$, for $m \in \mathbb{N}$.

An m-ary fractional operation ω on D is a probability distribution on $\mathcal{O}_D^{(m)}$. The support of ω is defined as $Supp(\omega) = \{g \in \mathcal{O}_D^{(m)} \mid \omega(g) > 0\}$.

A m-ary fractional operation ω on D with finite support is said to be a fractional polymorphism of a cost function f if, for any $x_1, x_2, \ldots, x_m \in D^n$, we have

$$\sum_{g \in Supp(\omega)} \omega(g) f(g(x_1, x_2, \ldots, x_m)) \leq \frac{1}{m}(f(x_1) + f(x_2) + \cdots + f(x_m)),$$

where g is applied componentwise.

For a valued constraint language Γ, $fPol(\Gamma)$ denotes the set of fractional operations that are fractional polymorphisms of every cost function in Γ.

Symmetric fractional polymorphisms

A (m-ary) fractional polymorphism is said to be symmetric if all operations g in its support is symmetric, i.e. for every permutation $\pi \in Sym(1, \ldots, m)$, we have $g(x_1, \ldots, x_m) = g(x_{\pi(1)}, \ldots, x_{\pi(m)})$.

Example: submoduar semilinear languages

Let the domain D be totally ordered. We say that $f : D^n \to \mathbb{Q}$ is submodular if for each $x, y \in D^n$

$$f(x) + f(y) \geq f(\max\{x, y\}) + f(\min\{x, y\})$$

Example: submoduar semilinear languages

Let the domain D be totally ordered. We say that $f : D^n \to \mathbb{Q}$ is submodular if for each $x, y \in D^n$

$$f(x) + f(y) \geq f(\max\{x, y\}) + f(\min\{x, y\})$$

Submodularity is an important concept in discrete optimisation.

A cost function is submodular iff it has the (binary) fractional polymorphism $\omega : \mathcal{O}_D^{(2)} \to [0, 1]$,

$$\omega(g) = \begin{cases} \frac{1}{2} & \text{if} \quad g = \max \\ \frac{1}{2} & \text{if} \quad g = \min \\ 0 & \text{if} \quad \text{otherwise} \end{cases}$$

Submodular semilinear VCSPs

Γ: submodular language over a totally ordered domain D.

- If D is finite then the VCSP is in P (Cohen, Cooper, Jeavons, Krokhin).
- What is the computational complexity of VCSP(Γ) if Γ is a submodular semilinear language?

Submodular semilinear functions

Examples of submodular semilinear cost functions:

- all unary cost functions are submodular;
- all linear cost functions are submodular;
- the cost function $f\colon \mathbb{Q}^3 \to \mathbb{Q}$, $f(x_1, x_2, x_3) = \max(x_1, x_2, x_3)$ is submodular;
- the cost function $f\colon \mathbb{Q}^2 \to \mathbb{Q}$, $f(x, y) = \min(x, -y)$ is submodular.

Submodular semilinear functions

Examples of submodular semilinear cost functions:

- all unary cost functions are submodular;
- all linear cost functions are submodular;
- the cost function $f \colon \mathbb{Q}^3 \to \mathbb{Q}$, $f(x_1, x_2, x_3) = \max(x_1, x_2, x_3)$ is submodular;
- the cost function $f \colon \mathbb{Q}^2 \to \mathbb{Q}$, $f(x, y) = \min(x, -y)$ is submodular.

Lemma (Topkis, 1978)

A binary function $f \colon \mathbb{Q}^2 \to \mathbb{Q}$ is submodular if, and only if, for every $\alpha_1 < \alpha_2$ and $\beta_1 < \beta_2$ in $\mathbb{Q}$

$$f(\alpha_1, \beta_1) + f(\alpha_2, \beta_2) \leq f(\alpha_1, \beta_2) + f(\alpha_2, \beta_1).$$

Submodular semilinear functions

Examples of submodular semilinear cost functions:

- all unary cost functions are submodular;
- all linear cost functions are submodular;
- the cost function $f\colon \mathbb{Q}^3 \to \mathbb{Q}, f(x_1, x_2, x_3) = \max(x_1, x_2, x_3)$ is submodular;
- the cost function $f\colon \mathbb{Q}^2 \to \mathbb{Q}, f(x, y) = \min(x, -y)$ is submodular.

Lemma (Topkis, 1978)

A binary function $f\colon \mathbb{Q}^2 \to \mathbb{Q}$ is submodular if, and only if, for every $\alpha_1 < \alpha_2$ and $\beta_1 < \beta_2$ in $\mathbb{Q}$

$$f(\alpha_1, \beta_1) + f(\alpha_2, \beta_2) \leq f(\alpha_1, \beta_2) + f(\alpha_2, \beta_1).$$

Theorem (Topkis, 1978)

$f\colon \mathbb{Q}^n \to \mathbb{Q}$ is submodular if and only if the (binary) projection to every plane parallel to one of the coordinate planes is submodular.

Submodular semilinear functions

A function $f\colon \mathbb{Q}^n \to \mathbb{Q}$ is separable if $f(x) = \sum_{i=1}^{n} f_i(x_i)$ for all $x = (x_1, \ldots, x_n)$, with $x_i \in \mathbb{Q}$ for $i = 1, \ldots, n$.

Submodular semilinear functions

A function $f : \mathbb{Q}^n \to \mathbb{Q}$ is separable if $f(x) = \sum_{i=1}^{n} f_i(x_i)$ for all $x = (x_1, \ldots, x_n)$, with $x_i \in \mathbb{Q}$ for $i = 1, \ldots, n$.

Theorem (Topkis, 1978)

If D_i is a chain (totally ordered set) for $i = 1, \ldots, n$, then f is separable on $\prod_{i=1}^{n} D_i$ if, and only if, both f and $-f$ are submodular on $\prod_{i=1}^{n} D_i$.

Submodular semilinear functions

A function $f : \mathbb{Q}^n \to \mathbb{Q}$ is separable if $f(x) = \sum_{i=1}^n f_i(x_i)$ for all $x = (x_1, \ldots, x_n)$, with $x_i \in \mathbb{Q}$ for $i = 1, \ldots, n$.

Theorem (Topkis, 1978)

If D_i is a chain (totally ordered set) for $i = 1, \ldots, n$, then f is separable on $\prod_{i=1}^n D_i$ if, and only if, both f and $-f$ are submodular on $\prod_{i=1}^n D_i$.

Proposition

$f_i : \mathbb{Q} \to \mathbb{Q}$, $i = 1, \ldots, n$ finitely many unary semilinear cost functions. Then, find $\inf_{x \in \mathbb{Q}}(f_1(x) + \cdots + f_n(x))$ is in P.
It follows that the VCSP for a language containing only separable semilinear cost functions is in P.

Submodular semilinear functions

Proposition

Maximum of non-decreasing unary functions are submodular.

Submodular semilinear functions

Proposition

Maximum of non-decreasing unary functions are submodular.

Example

$f\colon \mathbb{Q}^2 \to \mathbb{Q}, f(x_1, x_2) = \max(x_1 + 6, 3x_2)$ is submodular.

Submodular semilinear functions

Proposition

Maximum of non-decreasing unary functions are submodular.

Example

$f : \mathbb{Q}^2 \to \mathbb{Q}, f(x_1, x_2) = \max(x_1 + 6, 3x_2)$ is submodular.

Counterexample

$f : \mathbb{Q}^2 \to \mathbb{Q}, f(x_1, x_2) = \min(-x_1, -x_2 + 1)$. It is minimum of non-increasing functions and it is <u>not</u> submodular. Take, for instance $(2, 4), (5, -2) \in \mathbb{Q}^2$.

Submodular semilinear functions

Proposition

Minimum of a non-decreasing unary function and a non-increasing unary function are submodular.

Submodular semilinear functions

Proposition

Minimum of a non-decreasing unary function and a non-increasing unary function are submodular.

Example

$f : \mathbb{Q}^2 \to \mathbb{Q}, f(x_1, x_2) = \min(x_1 + 2, -x_2)$ is submodular.

Submodular semilinear functions

Proposition

Minimum of a non-decreasing unary function and a non-increasing unary function are submodular.

Example

$f : \mathbb{Q}^2 \to \mathbb{Q}, f(x_1, x_2) = \min(x_1 + 2, -x_2)$ is submodular.

Counterexample

$f : \mathbb{Q}^2 \to \mathbb{Q}, f(x_1, x_2) = \max(x_1, -x_2)$ is maximum of a non-decreasing function and a non-increasing function. It is not submodular: consider $(-3, 2), (5, 1) \in \mathbb{Q}^2$.

The expressive power

Let Γ be a valued constraint language.
A k-ary cost function f is expressible over Γ if there exists an instance $\mathcal{I}$ of $VCSP(\Gamma)$ with objective function $f_{\mathcal{I}}$ and with variables $V = \{x_1, \ldots, x_k, x_{k+1}, \ldots, x_n\}$, such that

$$f(x_1, \ldots, x_k) = \min_{x_{k+1}, \ldots, x_n} f_{\mathcal{I}}(x_1, \ldots, x_k, x_{k+1}, \ldots, x_n).$$

The expressive power

Let Γ be a valued constraint language.
A k-ary cost function f is expressible over Γ if there exists an instance $\mathcal{I}$ of $VCSP(\Gamma)$ with objective function $f_{\mathcal{I}}$ and with variables $V = \{x_1, \ldots, x_k, x_{k+1}, \ldots, x_n\}$, such that

$$f(x_1, \ldots, x_k) = \min_{x_{k+1}, \ldots, x_n} f_{\mathcal{I}}(x_1, \ldots, x_k, x_{k+1}, \ldots, x_n).$$

Expressive power of Γ: the set $\langle\Gamma\rangle$ of all cost functions expressible over Γ.

Remark: $\langle\Gamma\rangle$ is the closure of Γ under addition, non-negative scalar multiplication, minimisation over extra variables.

The expressive power

Let Γ be a valued constraint language.
A k-ary cost function f is expressible over Γ if there exists an instance $\mathcal{I}$ of $VCSP(\Gamma)$ with objective function $f_{\mathcal{I}}$ and with variables $V = \{x_1, \ldots, x_k, x_{k+1}, \ldots, x_n\}$, such that

$$f(x_1, \ldots, x_k) = \min_{x_{k+1}, \ldots, x_n} f_{\mathcal{I}}(x_1, \ldots, x_k, x_{k+1}, \ldots, x_n).$$

Expressive power of Γ: the set $\langle \Gamma \rangle$ of all cost functions expressible over Γ.

Remark: $\langle \Gamma \rangle$ is the closure of Γ under addition, non-negative scalar multiplication, minimisation over extra variables.

Proposition (Cohen, Cooper, Jeavons, 2006)

Γ *valued constraint language over a finite domain. Then* $fPol(\Gamma) = fPol(\langle \Gamma \rangle)$.

The proof works also for finite languages Γ over an infinite domain.

Corollary

Let Γ *be a semilinear language made up by:*

- *separable cost functions: that can be written as sum of unary cost functions;*

Corollary

Let Γ *be a semilinear language made up by:*

- *separable cost functions: that can be written as sum of unary cost functions;*
- $f(x) = \max\{f_1(x_1), \ldots, f_m(x_m)\}$, *where* $f_i(x_i)$ *are non-decreasing cost functions;*

Corollary

Let Γ *be a semilinear language made up by:*

- *separable cost functions: that can be written as sum of unary cost functions;*
- $f(x) = \max\{f_1(x_1), \ldots, f_m(x_m)\}$, *where* $f_i(x_i)$ *are non-decreasing cost functions;*
- $f(x) = \min\{f_1(x_1), f_2(x_2)\}$, *where* $f_1(x_1)$ *is a non-decreasing cost function and* $f_2(x_2)$ *is a non-increasing cost function;*

Corollary

Let Γ *be a semilinear language made up by:*

- *separable cost functions: that can be written as sum of unary cost functions;*
- $f(x) = \max\{f_1(x_1), \ldots, f_m(x_m)\}$*, where* $f_i(x_i)$ *are non-decreasing cost functions;*
- $f(x) = \min\{f_1(x_1), f_2(x_2)\}$*, where* $f_1(x_1)$ *is a non-decreasing cost function and* $f_2(x_2)$ *is a non-increasing cost function;*
- *(non-negative linear) combinations of previous cases.*

Corollary

Let Γ *be a semilinear language made up by:*

- *separable cost functions: that can be written as sum of unary cost functions;*
- $f(x) = \max\{f_1(x_1), \ldots, f_m(x_m)\}$, *where* $f_i(x_i)$ *are non-decreasing cost functions;*
- $f(x) = \min\{f_1(x_1), f_2(x_2)\}$, *where* $f_1(x_1)$ *is a non-decreasing cost function and* $f_2(x_2)$ *is a non-increasing cost function;*
- *(non-negative linear) combinations of previous cases.*

Then Γ *is a semilinear submodular language.*

Corollary

Let Γ *be a semilinear language made up by:*

- *separable cost functions: that can be written as sum of unary cost functions;*
- $f(x) = \max\{f_1(x_1), \ldots, f_m(x_m)\}$, *where* $f_i(x_i)$ *are non-decreasing cost functions;*
- $f(x) = \min\{f_1(x_1), f_2(x_2)\}$, *where* $f_1(x_1)$ *is a non-decreasing cost function and* $f_2(x_2)$ *is a non-increasing cost function;*
- *(non-negative linear) combinations of previous cases.*

Then Γ *is a semilinear submodular language.*

Γ is a tame submodular semilinear language if it satisfies the hypothesis of the corollary above.

Algorithm

Consider the following objective function

$$\Phi(x, y, z) = f_1(x) + f_2(y) + \max(g_1(y), g_2(z)) + \min(h_1(x), h_2(z)).$$

Algorithm

Consider the following objective function

$$\Phi(x, y, z) = f_1(x) + f_2(y) + \max(g_1(y), g_2(z)) + \min(h_1(x), h_2(z)).$$

Where the elementary unary functions are:

$$f_1(x) = \begin{cases} 5x + 2 & x < 4 \\ 1 & x = 4 \\ 2x - 5 & x > 4 \end{cases} \qquad f_2(y) = \begin{cases} -3y + 1 & y < -7 \\ -8 & y = -7 \\ y - 2 & y > -7 \end{cases}$$

$$g_1(y) = \begin{cases} 2y + 2 & y < 0 \\ 3 & y = 0 \\ y + 3 & y > 0 \end{cases} \qquad g_2(z) = \begin{cases} z + 1 & z < 2 \\ 3 & z = 2 \\ 2z - 1 & 2 < z < 3 \\ 7 & z = 3 \\ 2z + 3 & z > 3 \end{cases}$$

$$h_1(x) = \begin{cases} x - 3 & x < -1 \\ 0 & x = -1 \\ x + 2 & x > -1 \end{cases} \qquad h_2(z) = -z$$

Algorithm

- Define $B = \{-7, -1, 0, 2, 3, 4\}$ (special points).

- $(\mathbb{Q} \times E; \preceq)$, where $E = \{-1, 0, 1\}$ and $(a, b) \preceq (c, d)$ iff $a < c$ or $a = c$ and $b \leq d$.

Algorithm

- Define $B = \{-7, -1, 0, 2, 3, 4\}$ (special points).

- $(\mathbb{Q} \times E; \preceq)$, where $E = \{-1, 0, 1\}$ and $(a, b) \preceq (c, d)$ iff $a < c$ or $a = c$ and $b \leq d$.

- $\tilde{f}_1(x, \alpha) = \begin{cases} 5x + 2 & (x, \alpha) \prec (4, 0) \\ 1 & (x, \alpha) = (4, 0) \\ 2x - 5 & (x, \alpha) \succ (4, 0) \end{cases}$ $\tilde{f}_2(y, \alpha) = \begin{cases} -3y + 1 & (y, \alpha) \prec (-7, 0) \\ -8 & (y, \alpha) = (-7, 0) \\ y - 2 & (y, \alpha) \succ (-7, 0) \end{cases}$

$$\tilde{g}_1(y, \alpha) = \begin{cases} 2y + 2 & (y, \alpha) \prec (0, 0) \\ 3 & (y, \alpha) = (0, 0) \\ y + 3 & (y, \alpha) \succ (0, 0) \end{cases} \quad \tilde{g}_2(z, \alpha) = \begin{cases} z + 1 & (z, \alpha) \prec (2, 0) \\ 3 & (z, \alpha) = (2, 0) \\ 2z - 1 & (2, 0) \prec (z, \alpha) \prec (3, 0) \\ 7 & (z, \alpha) = (3, 0) \\ 2z + 3 & (z, \alpha) \succ (3, 0) \end{cases}$$

$$\tilde{h}_1(x, \alpha) = \begin{cases} x - 3 & (x, \alpha) \prec (-1, 0) \\ 0 & (x, \alpha) = (-1, 0) \\ x + 2 & (x, \alpha) \succ (-1, 0) \end{cases} \quad \tilde{h}_2(z, \alpha) = -z$$

Algorithm

- Every $\tilde{f}$ is unary and inherits the monotonicity of f, therefore

  $$\tilde{\Phi}((x,\alpha),(y,\beta),(z,\gamma)) = \tilde{f}_1(x,\alpha)+\tilde{f}_2(y,\beta)+\max(\tilde{g}_1(y,\beta),\tilde{g}_2(z,\gamma))+\min(\tilde{h}_1(x,\alpha),\tilde{h}_2(z,\gamma))$$

  is an instance of a VCSP for a new tame submodular semilinear language, Γ' over $\mathbb{Q} \times E$.

Algorithm

- Every $\tilde{f}$ is unary and inherits the monotonicity of f, therefore

  $$\tilde{\Phi}((x,\alpha),(y,\beta),(z,\gamma)) = \tilde{f}_1(x,\alpha)+\tilde{f}_2(y,\beta)+\max(\tilde{g}_1(y,\beta),\tilde{g}_2(z,\gamma))+\min(\tilde{h}_1(x,\alpha),\tilde{h}_2(z,\gamma))$$

  is an instance of a VCSP for a new tame submodular semilinear language, Γ' over $\mathbb{Q} \times E$.
- $\inf_{\mathbb{Q}} \Phi = \inf_{\mathbb{Q}\times E} \tilde{\Phi}$.

Algorithm

- Every $\tilde{f}$ is unary and inherits the monotonicity of f, therefore

  $$\tilde{\Phi}((x,\alpha),(y,\beta),(z,\gamma)) = \tilde{f}_1(x,\alpha)+\tilde{f}_2(y,\beta)+\max(\tilde{g}_1(y,\beta),\tilde{g}_2(z,\gamma))+\min(\tilde{h}_1(x,\alpha),\tilde{h}_2(z,\gamma))$$

  is an instance of a VCSP for a new tame submodular semilinear language, Γ' over $\mathbb{Q} \times E$.
- $\inf_{\mathbb{Q}} \Phi = \inf_{\mathbb{Q}\times E} \tilde{\Phi}$.
- $D = \{(\alpha, 0), (\alpha, -1), (\alpha, 1) \mid \alpha \in B\}$ (finite).

Algorithm

- Every $\tilde{f}$ is unary and inherits the monotonicity of f, therefore

  $$\tilde{\Phi}((x,\alpha),(y,\beta),(z,\gamma)) = \tilde{f_1}(x,\alpha)+\tilde{f_2}(y,\beta)+\max(\tilde{g_1}(y,\beta),\tilde{g_2}(z,\gamma))+\min(\tilde{h_1}(x,\alpha),\tilde{h_2}(z,\gamma))$$

  is an instance of a VCSP for a new tame submodular semilinear language, Γ' over $\mathbb{Q} \times E$.
- $\inf_{\mathbb{Q}} \Phi = \inf_{\mathbb{Q}\times E} \tilde{\Phi}$.
- $D = \{(\alpha, 0), (\alpha, -1), (\alpha, 1) \mid \alpha \in B\}$ (finite).
- Cost functions in Γ' are still submodular over $D \subset \mathbb{Q} \times E$.

Algorithm

- Every $\tilde{f}$ is unary and inherits the monotonicity of f, therefore

  $$\tilde{\Phi}((x,\alpha),(y,\beta),(z,\gamma)) = \tilde{f}_1(x,\alpha)+\tilde{f}_2(y,\beta)+\max(\tilde{g}_1(y,\beta),\tilde{g}_2(z,\gamma))+\min(\tilde{h}_1(x,\alpha),\tilde{h}_2(z,\gamma))$$

  is an instance of a VCSP for a new tame submodular semilinear language, Γ' over $\mathbb{Q} \times E$.
- $\inf_{\mathbb{Q}} \Phi = \inf_{\mathbb{Q}\times E} \tilde{\Phi}$.
- $D = \{(\alpha, 0), (\alpha, -1), (\alpha, 1) \mid \alpha \in B\}$ (finite).
- Cost functions in Γ' are still submodular over $D \subset \mathbb{Q} \times E$.

Fact

Γ tame submodular semilinear language.
If $\inf_{\mathbb{Q}\times E} \tilde{\Phi} = \inf_D \tilde{\Phi}$, then there exists a polynomial-time reduction from a VCSP(Γ) to a VCSP for a submodular language over a finite domain.
In particular, VCSP(Γ) is in P.

Next steps and open problems

1. Prove that $\inf_{\mathbb{Q}\times E} \tilde{\Phi} = \inf_D \tilde{\Phi}$.
2. Adapt the algorithm to the case in which all elementary unary cost function in the instance are linear (no special points).
3. Find a syntactic characterisation for all submodular semilinear functions.
4. Does $fPol(\Gamma) = fPol(\Delta)$ implies $\langle \Gamma \rangle = \langle \Delta \rangle$?

Thank you