Теоретические проблемы обработки и распознавания образов и речи

УДК 519.683:681.3

М.И. Шлезингер

Распознавание образов как реализация определенного подкласса процессов мышления

Представлен обзор исследований проблемы распознавания образов, выполненных в течение последнего двадцатилетия в Международном научно-учебном центре информационных технологий и систем с момента его основания. Это – исследования на стыке классической проблемы распознавания и проблемы совместимости системы ограничений, известной как *Constraint Satisfaction Problem*. Система понятий, задач и алгоритмов на стыке этих направлений формализует определенный тип мыслительных процессов, осознанно или неосознанно выполняемых человеком или другими живыми существами.

Подано огляд досліджень проблеми розпізнавання образів, виконаних протягом останнього двадцятиріччя у Міжнародному науково-навчальному центрі інформаційних технологій та систем з дня його заснування. Це – дослідження на перетині класичної проблеми розпізнавання і проблеми несуперечності системи обмежень, відомої як *Constraint Satisfaction Problem*. Система понять, задач і алгоритмів на перетині цих напрямів формалізує певний тип процесів мислення, що свідомо чи несвідомо виконує людина або інші живі істоти.

Введение. Исследования, проведенные в Международном научно-учебном центре информационных технологий и систем за последние два десятилетия, находятся на стыке двух научных направлений, существовавших до обозреваемого периода независимо друг от друга. Первое направление состоит в формулировке задач распознавания и их решении на основе порождающих моделей множества распознаваемых сигналов. Основополагающие идеи этого направления сформулированы в работах В.А. Ковалевского и Т.К. Винцюка [1–6] 1967–1987 годов. Второе направление составляют исследования проблемы совместимости системы ограничений, известной как *Constraint Satisfaction Problem* [7]. На стыке этих двух направлений сформировалась новая система понятий, задач и алгоритмов, которая образует одну из парадигм машинного образного мышления.

Вычисление и мышление как две различные схемы решения задач

Будем различать мыслительные и вычислительные процессы на основе различия между двумя форматами представления задачи для ее решения на искусственно созданном устройстве. Для решения задачи на вычислительном устройстве ее следует представить в виде тройки $\langle X, Y, f \rangle$, где X и Y – множества исходных данных и, соответственно, результатов решения, а $f: X \to Y$ – алгоритм вычисления результата $f(x) \in Y$ для исходных данных $x \in X$. В указанной тройке отсутствует то, что в обыденной практике понимается как условие задачи, подлежащей решению. Дополним тройку $\langle X, Y, f \rangle$ функцией $\varphi: X \times Y \to \{0,1\}$, формализующей понятие условия задач и для каждой пары $x \in X, y \in Y$ указывающей, является ли y правильным решением при исходных данных x. Если на вход вычислительного устройства поступает не только тройка $\langle X, Y, f \rangle$, но и условие φ задачи, вычислительное устройство приобретает возможность самоконтроля получаемых результатов, что в свою очередь позволяет смягчить требования к алгоритму f. Последний может иметь эвристический характер, т.е. гарантировать решение задачи не для любых исходных данных, а только для некоторого, трудно распознаваемого подмножества. Самоконтроль $\varphi: X \times Y \to \{0,1\}$ позволяет блокировать подачу на выход ошибочного решения.

Мыслительное устройство решает задачу не на основании тройки $\langle X, Y, f \rangle$, а на основании тройки $\langle X, Y, \varphi: X \times Y \to \{0,1\} \rangle$, где φ – усло-

вие задачи. В отличие от вычисления, состоящего в получении решения $y^* = f(x^*)$ для данных x^* с помощью известного алгоритма f, мышление состоит в том, что для заданного x^* и условия $\varphi: X \times Y \to \{0,1\}$ выполняется поиск решения y^*, такого, что $\varphi(x^*, y^*) = 1$. Теоретический анализ процессов мышления основан на фундаментальных понятиях теории распознавания совместимости или противоречивости системы ограничений [7]. Применение этой теории к задачам распознавания образов, особенно, к распознаванию изображений, оказалось плодотворным как для практики машинного зрения, так и для самой науки о совместимости ограничений. В практике машинного зрения произошел прорыв в таких традиционно трудных задачах, как стереозрение и текстурная сегментация [8–13]. Вместе с тем, соприкосновение математической теории совместимости ограничений с практическими задачами машинного зрения привело к существенному расширению математической проблематики этой науки. Прежде всего это расширение состояло в задании той или иной функции качества на множестве решений системы ограничений [14–18]. Задача в этом случае заключается в поиске не произвольного решения системы ограничений, а наилучшего решения, оптимизирующего заданную функцию качества. Следующий этап обобщения состоит в том, что для заданной системы ограничений и функции качества на множестве ее решений требуется найти не единственное ее решение, пусть и наилучшее, а заданное количество наилучших решений [19].

Таким образом, на стыке проблем распознавания совместимости ограничений и распознавания образов сформировалось новое направление в моделировании определенного класса мыслительных процессов. Размеры журнальной публикации, конечно, не достаточны для исчерпывающего обзора того мощного потока исследований во всем мире, которые сыграли ключевую роль в формировании этого направления. Учитывая юбилейный характер данного выпуска журнала, статья показывает лишь вклад МНУЦИТиС в процесс его становления.

Определение основных понятий

Пусть T и K – два конечных множества, элементы которых назовем *объектами* и *метками*. Структурой множества T назовем некоторое подмножество $\mathbb{S} \subset T \times T$ пар (i, j) объектов $i \in T$, $j \in T$. Для любого подмножества $S \subseteq T$ обозначим K^S множество всех возможных отображений вида $S \to K$.

Пусть $\bar{x}: T \to K$ – функция, называемая *разметкой*. Значение разметки $\bar{x}: T \to K$ для объекта $i \in T$ обозначим x_i, а ее сужение на подмножество $S \subset T$ обозначим x_S. В случае, когда требуется подчеркнуть, что $S = P \cup R$, $P \cap R = \varnothing$, сужение $\bar{x}$ на S будет записано не в виде $x_{P \cup R}$, а в виде пары (x_P, x_R). Сужение $\bar{x}$ на $\{i, j\}$ обозначим (x_i, x_j).

Пусть $(W, \oplus, \otimes)$ – коммутативное полукольцо и для каждой пары $(i, j) \in \mathbb{S}$ определена функция $\varphi_{ij}: K \times K \to W$.

Определение 1. Исходные данные $(\oplus,\otimes)$-задачи разметки – это пятерка

$$\Phi = \langle T, K, (W, \oplus, \otimes), \mathbb{S}, (\varphi_{ij} \mid (i, j) \in \mathbb{S}) \rangle, \quad (1)$$

а решение $(\oplus,\otimes)$-задачи – значение

$$w^* = \bigoplus_{\bar{x} \in K^T} \bigotimes_{(i,j) \in \mathbb{S}} \varphi_{ij}(x_i, x_j) \in W \quad (2)$$

и разметка $\bar{x}^* \in K^T$, такая, что

$$\bigotimes_{(i,j) \in \mathbb{S}} \varphi_{ij}(x_i^*, x_j^*) = \bigoplus_{\bar{x} \in K^T} \bigotimes_{(i,j) \in \mathbb{S}} \varphi_{i,j}(x_i, x_j) = w^*, \quad (3)$$

если таковая существует. ■

Задача (1–3), сформулированная на полукольце $(\{0,1\}, \vee, \wedge)$, соответствует классической задаче распознавания совместимости ограничений [7]. Задача распознавания совместимости размытых ограничений [20] есть задача (1–3), сформулированная на полукольце $(W, \max, \min)$, где W – некоторое упорядоченное множество. Оптимизационная задача разметки [14–18] – это задача (1–3), сформулированная на полукольце $(W, \min, \otimes)$, где $\otimes$ – любая операция, образующая с операцией min полукольцо на упорядоченном множестве W. В дальнейшем используем обозначения $(\vee, \wedge)$-за-

дача или (max,min)-задача, или (min,+)-задача в зависимости от того, в каком полукольце эта задача сформулирована, или обозначение $(\oplus,\otimes)$-задача, если она сформулирована в произвольном, но заданном полукольце.

Задачи разметки на ациклических структурах

Множество T вместе со структурой $\mathbb{S}$ второго порядка образуют неориентированный граф, множеством вершин которого является T, а вершины $i \in T$ и $j \in T$ соединены ребром тогда и только тогда, когда $ij \in \mathbb{S}$. Структуру $\mathbb{S}$ множества T назовем *ациклической*, если пара $(T,\mathbb{S})$ образует связный граф, не содержащий циклов. Для этого случая, с чисто техническими целями видоизменим выражение (2) для искомого решения $(\oplus,\otimes)$-задачи, включив в него сомножители, каждый из которых зависит только от одной переменной. Выражение (2) в этом случае приобретает вид

$$w^* = \bigoplus_{\bar{x}\in K^T}\left[\bigotimes_{i\in T}\varphi_i(x_i)\otimes\bigotimes_{ij\in\mathbb{S}}\varphi_{ij}(x_i,x_j)\right]\in W. \quad (4)$$

Ациклическая задача (4) эффективно решается в общем виде для любого полукольца $(W,\oplus,\otimes)$, и это решение состоит в последовательном исключении объектов из T посредством следующей процедуры динамического программирования [18, 21]. Выбирают объект $l \in T$, для которого существует один и только один объект $m \in T$, такой, что $\{m,l\} \in \mathbb{S}$. Для выбранного объекта l определяют множество $T^* = T \setminus \{l\}$, структуру $\mathbb{S}^* = \mathbb{S} \setminus \{\{m,l\}\}$ и функции $\psi_i : K \to W$, $i \in T^*$, так, что

$$\psi_m(x) = \varphi_m(x)\otimes\bigoplus_{y\in K}[\varphi_{ml}(x,y)\otimes\varphi_l(y)]$$
$$\text{и} \qquad \psi_i(x) = \varphi_i(x) \text{ для } i \neq m. \quad (5)$$

Результатом исключения объекта $l \in T$ является $(\oplus,\otimes)$-задача вычисления значения

$$u^* = \bigoplus_{\bar{x}\in K^{T^*}}\left[\bigotimes_{i\in T^*}\psi_i(x_i)\otimes\bigotimes_{ij\in\mathbb{S}^*}\varphi_{ij}(x_i,x_j)\right]\in W, \quad (6)$$

совпадающего с искомым решением (4) исходной задачи, так как

$$w^* = \bigoplus_{\bar{x}\in K^T}\left[\bigotimes_{i\in T}\varphi_i(x_i)\otimes\bigotimes_{ij\in\mathbb{S}}\varphi_{ij}(x_i,x_j)\right] =$$
$$= \bigoplus_{\bar{x}\in K^{T^*}}\bigoplus_{x_l\in K}\Big[\varphi_l(x_l)\otimes\varphi_{ml}(x_m,x_l)\otimes$$
$$\otimes\bigotimes_{i\in T^*}\varphi_i(x_i)\otimes\bigotimes_{ij\in\mathbb{S}^*}\varphi_{ij}(x_i,x_j)\Big] =$$
$$= \bigoplus_{\bar{x}\in K^{T^*}}\bigotimes_{i\in T^*}\varphi_i(x_i)\otimes\bigotimes_{ij\in\mathbb{S}^*}\varphi_{ij}(x_i,x_j)\otimes$$
$$\otimes\left[\bigoplus_{x_l\in K}\varphi_l(x_l)\otimes\varphi_{ml}(x_m,x_l)\right] =$$
$$= \bigoplus_{\bar{x}\in K^{T^*}}\left[\bigotimes_{i\in T^*}\psi_i(x_i)\otimes\bigotimes_{ij\in\mathbb{S}^*}\varphi_{ij}(x_i,x_j)\right] = u^*.$$

Как видно из (5), исключение одного объекта имеет сложность $|K|^2$-кратного выполнения операций $\oplus$ и $\otimes$, а решение $(\oplus,\otimes)$-задачи в целом – сложность $(|T|\cdot|K|^2)$-кратного выполнения этих операций.

Множество $(\oplus,\otimes)$-задач на ациклических структурах допускает единообразное решение с помощью общего алгоритма, который за время порядка $(|T|\cdot|K|^2)$ решает задачу для любых заданных функций φ_{ij} и любого заданного полукольца $(\oplus,\otimes)$. Для задач на произвольных структурах это единообразие уже не наблюдается: задачи на разных полукольцах решаются различными, хотя в чем-то и сходными алгоритмами.

Разрешимые подклассы $(\vee,\wedge)$-задач

Языки полиномиально разрешимых подклассов $(\vee,\wedge)$-задач [22]. Ради упрощения вида формул и без потери общности положим, что структура $\mathbb{S}$ состоит из всех возможных пар $i,j \in T$, $i \neq j$, и при этом $\varphi_{ij}(k,k') = \varphi_{ji}(k',k)$. В таком случае исходные данные $(\vee,\wedge)$-задачи можно представить не в виде пятерки (1), а более лаконично – в виде тройки

$$\Phi = \langle T, K, (\varphi_{ij} : K\times K \to \{0,1\} \mid i,j \in T)\rangle. \quad (7)$$

Собственно $(\vee,\wedge)$-задача заключается в вычислении значения

$$w^* = \bigvee_{\bar{x}\in K^T} \bigwedge_{i,j\in T} \varphi_{ij}(x_i, x_j) \in \{0,1\}, \qquad (8)$$

и, если $w^* = 1$, отыскании разметки $\bar{x}^* : T \to K$, такой, что $\varphi_{ij}(x_i^*, x_j^*) = 1$ для всех $i \in T, j \in T$. В дальнейшем рассмотрим произвольные, но фиксированные множества T и K и поэтому, как правило, они не будут упоминаться. Кроме того, не будет указываться формат функций φ_{ij}, если это ясно из контекста. Так, выражение (7) будет кратко представлено в виде $\Phi = (\varphi_{ij} : K \times K \to \to \{0,1\} \mid i, j \in T)$ или еще короче $\Phi = (\varphi_{ij} \mid i, j \in T)$.

Множество задач, сформулированных в формате (7, 8), образует NP-полный класс. Полиномиально разрешимые подклассы этого класса формируют путем ограничений класса функций φ_{ij} в исходных данных задачи (7). В теории распознавания совместимости ограничений это сужение выполняют на основе математических понятий [22], которые сформулируем на адекватном для данной статьи уровне общности.

Пусть $\varphi : K \times K \to \{0,1\}$ – функция двух переменных, а p – двухместный оператор вида $K \times K \to K$ или трехместный оператор вида $K \times K \times K \to K$.

Определение 2. Функция $\varphi : K \times K \to \{0,1\}$ называется *инвариантом двухместного оператора* $p : K \times K \to K$, а оператор p *– полиморфизмом* функции φ, если для любых двух пар $x, x' \in K$ и $y, y' \in K$ справедлива импликация $\varphi(x, x') \wedge \varphi(y, y') \to \varphi(p(x, y), p(x', y'))$.

Функция $\varphi : K \times K \to \{0,1\}$ называется *инвариантом трехместного оператора* $p : K \times K \times \times K \to K$, а оператор p – полиморфизмом функции φ, если для любых трех пар $x, x' \in K$, $y, y' \in K$ и $z, z' \in K$ справедлива импликация $\varphi(x, x') \wedge \varphi(y, y') \wedge \varphi(z, z') \to \varphi(p(x, y, z), p(x', y', z'))$. ■

Определение 3. Пусть Γ – подмножество функций вида $\varphi : K \times K \to \{0,1\}$, называемое *языком* задач. Задача (7, 8) формулируется в языке Γ, если $\varphi_{ij} \in \Gamma$ для всех $i, j \in T$. ■

Определение 4. Ассоциативный, коммутативный, идемпотентный оператор $p : K \times K \to K$ называется оператором полурешетки на множестве K.

Оператор $p : K \times K \times K \to K$ называется мажоритарным, если для любых $x, y \in K$ выполняется равенство $p(x, x, y) = p(x, y, x) = p(y, x, x) = x$.

Оператор $p : K \times K \times K \to K$ называется оператором Мальцева, если для любых $x, y \in K$ выполняется равенство $p(x, x, y) = p(y, x, x) = y$. ■

В следующей теореме сформулирован определенный подкласс полиномиально разрешимых $(\vee, \wedge)$-задач [22].

Теорема 1. Пусть Γ – язык задач, для которого выполняется по крайней мере одно из следующих трех условий:

- существует по крайней мере один оператор полурешетки, относительно которого инвариантны все функции из Γ;
- существует по крайней мере один мажоритарный оператор, относительно которого инвариантны все функции из Γ;
- существует по крайней мере один оператор Мальцева, относительно которого инвариантны все функции из Γ.

В таком случае множество $(\vee, \wedge)$-задач, формулируемых в этом языке, образует полиномиально разрешимый подкласс. ■

В следующем подразделе сформулируем другой полиномиально разрешимый подкласс $(\vee, \wedge)$-задач. Он пересекается с подклассом задач, сформулированным в Теореме 1, но не является его подмножеством: сформулированный далее полиномиально разрешимый подкласс содержит в себе и задачи, не входящие в приведенный подкласс. Идейную основу этого подкласса составляет понятие согласованных $(\vee, \wedge)$-задач, первоначально сформулированное в работах [16, 17] и отшлифованное в многочисленных исследованиях, обзор которых представлен в [7].

Согласованные $(\vee, \wedge)$-задачи и процедуры вычеркивания

Определение 5. Задача $\Phi = (\varphi_{ij} : K \times K \to \to \{0,1\} \mid i, j \in T)$ называется пустой, если

$\varphi_{ij}(x_i, x_j) = 0$ для любых двух объектов $i, j \in T$ и любых двух меток $x_i, x_j \in K$. ■

Определение 6. Задачи $\Phi = (\varphi_{ij} \mid i, j \in T)$ и $\Phi^* = (\varphi^*_{ij} \mid i, j \in T)$ эквивалентны, если $\bigwedge\limits_{i,j \in T} \varphi_{ij}(x_i, x_j) = \bigwedge\limits_{i,j \in T} \varphi^*_{ij}(x_i, x_j)$ для любой разметки $\bar{x} \in K^T$. ■

Определение 7. Задача $\Phi = (\varphi_{ij} : K \times K \to \to \{0,1\} \mid i, j \in T)$ называется согласованной, если

$$\varphi_{ij}(x_i, x_j) = \varphi_{ij}(x_i, x_j) \wedge \wedge \bigvee_{x_l \in K} [\varphi_{il}(x_i, x_l) \wedge \varphi_{lj}(x_l, x_j)]$$

для любых трех объектов $i, j, l \in T$ и любых двух меток $x_i, x_j \in K$. ■

Для любой $(\vee, \wedge)$-задачи Φ существует согласованный эквивалент Φ^*, называемый ядром задачи Φ, которое строят путем многократного выполнения операции, называемой *вычеркиванием*. Один акт вычеркивания состоит в поиске трех объектов $i, j, l \in T$ и двух меток $x_i, x_j \in K$, нарушающих согласованность задачи, т.е. таких, что $\varphi_{ij}(x_i, x_j) = 1$, а $\bigvee\limits_{x_l \in K} [\varphi_{il}(x_i, x_l) \wedge \varphi_{lj}(x_l, x_j)] = 0$. Если такие тройка $i, j, l \in T$ и пара $x_i, x_j \in K$ найдены, то прежнее значение $\varphi_{ij}(x_i, x_j) = 1$ заменяют новым значением $\varphi_{ij}(x_i, x_j) = 0$. Замена значения $\varphi_{ij}(x_i, x_j)$ называется *вычеркиванием* четверки (i, j, x_i, x_j). После конечного количества вычеркиваний этот процесс завершается получением пустого или непустого ядра исходной задачи. Очевидно, что пустота ядра задачи – достаточное, но не необходимое условие ее противоречивости, и, аналогично, непустота ядра – необходимое, но не достаточное условие совместимости условий задачи. Однако на определенном подклассе задач эти условия необходимы и достаточны. Этот подкласс формулируется на основании понятий, близких к рассмотренным в предыдущем подразделе, но все же отличающихся от них.

Пусть для каждого $i \in T$ определен оператор $p_i : K \times K \times K \to K$.

Определение 8. Задачу $\Phi = (\varphi_{ij} : K \times K \to \to \{0,1\} \mid i, j \in T)$ назовем инвариантной относительно совокупности $P = (p_i \mid i \in T)$ мажоритарных операторов, если для любых двух объектов $i, j \in T$ и любых трех пар $x, x' \in K$, $y, y' \in K$, $z, z' \in K$ меток справедлива импликация

$$\varphi_{ij}(x, x') \wedge \varphi_{ij}(y, y') \wedge \varphi_{ij}(z, z') \to \to \varphi_{ij}(p_i(x, y, z), p_j(x', y', z')).$$ ■

Пусть для каждого $i \in T$ определен оператор $p_i : K \times K \to K$.

Определение 9. Задачу $\Phi = (\varphi_{ij} : K \times K \to \to \{0,1\} \mid i, j \in T)$ назовем инвариантной относительно совокупности $P = (p_i \mid i \in T)$ операторов, образующих полурешетку, если для любых двух объектов $i, j \in T$ и любых двух пар $x, x' \in K$, $y, y' \in K$ меток справедлива импликация $\varphi_{ij}(x, x') \wedge \varphi_{ij}(y, y') \to \varphi_{ij}(p_i(x, y), p_j(x', y'))$. ■

Определения 8 и 9 расширяют понятия полиморфизмов и инвариантов, принятые в теории $(\vee, \wedge)$-задач и сформулированные в Определении 2. В теории $(\vee, \wedge)$-задач полагают, что оператор $P = (p_i \mid i \in T)$ однороден на множестве T в том смысле, что его компоненты p_i не зависят от $i \in T$. В соответствии с Определениями 8 и 9 каждому $i \in T$ соответствует свой оператор p_i и в этом смысле оператор $P = (p_i \mid i \in T)$ может быть неоднородным.

Везде в дальнейшем будем говорить, что для задачи Φ существует подходящий полиморфизм, если существует совокупность $P = (p_i \mid i \in T)$ мажоритарных операторов, относительно которой задача Φ инвариантна, или если существует совокупность операторов, образующих полурешетку, относительно которой эта задача инвариантна.

Теорема 2. Пусть для задачи $\Phi = (\varphi_{ij} : K \times \times K \to \{0,1\} \mid i, j \in T)$ существует подходящий полиморфизм. В таком случае условия задачи Φ противоречивы тогда и только тогда, когда ядро $\Phi^* = (\varphi^*_{ij} : K \times K \to \{0,1\} \mid i, j \in T)$ этой задачи пустое. ■

Теорема доказывается сравнительно легко для случая, когда компонентами p_i подходящего полиморфизма $P = (p_i \mid i \in T)$ являются операторы, образующие на множестве K полурешетку. Доказательство теоремы несколько сложнее для случая, когда подходящий для задачи полиморфизм составлен из мажоритарных операторов. Справедливость теоремы для этого случая следует из результатов, описанных в [23] и относящихся к решению (max, min)-задач.

Из теоремы 2 следует простой алгоритм проверки совместимости условий в задачах с подходящим полиморфизмом. Проверка совместимости в этом случае сводится к построению ядра Φ^* исходной задачи Φ и проверке, что полученное ядро Φ^* не пустое. Как видно, этот алгоритм можно реализовать, даже не зная подходящий оператор P, относительно которого инвариантна решаемая задача. Достаточно лишь гарантии, что такой оператор существует. Поэтому указанный алгоритм непосредственно применим, если такая априорная гарантия следует, например, из специфики ее прикладного содержания. Если же такой гарантии нет, то предлагаемый алгоритм следует дополнить средствами самоконтроля с целью блокирования ошибочного решения задач, для которых подходящий полиморфизм отсутствует.

Алгоритм решения $(\vee,\wedge)$-задач с самоконтролем

Определение 10. Сечением $(\vee,\wedge)$-задачи $\Phi = (\varphi_{ij} \mid i, j \in T)$ по подмножеству $S \subseteq T$ и разметке $\bar{x}^* : S \to K$ называется задача $\Phi^* = (\varphi^*_{ij} \mid i, j \in T)$, такая, что $\varphi^*_{ij}(x_i, x_j) = 0$, если $i \in S$ и $x_i \neq x^*_i$, и $\varphi^*_{ij}(x_i, x_j) = \varphi_{ij}(x_i, x_j)$ для всех прочих $i, j \in T$, $x_i, x_j \in K$. ■

Как видно из Определения 10, сечением задачи Φ по пустому подмножеству является сама задача Φ. Из Определения 10 следует также справедливость леммы.

Лемма 1. Если сечение задачи по некоторому непустому подмножеству $S \subseteq T$ и разметке $\bar{x}^* : S \to K$ имеет непустое ядро, то $\bigwedge\limits_{i,j \in S} \varphi_{ij}(x^*_i, x^*_j) = 1$. ■

Примем для удобства, что $T = \{1,2,\ldots,n\}$, и определим семейство вложенных друг в друга подмножеств $S_r \subset T$, $r \in \{0,1,2,\ldots,n\}$, таких, что $S_0 = \varnothing$, $S_r = S_{r-1} \cup \{r\}$ и $S_n = T$. В силу свойств задач, для которых существует подходящий полиморфизм (см. Теорему 2), справедлива также лемма.

Лемма 2. Пусть $\Phi = (\varphi_{ij} : K \times K \to \{0,1\} \mid i, j \in T)$ – задача, для которой существует подходящий полиморфизм. Пусть для некоторого $r \in \{2,\ldots,n\}$ существует разметка $\bar{x}_{r-1} : S_{r-1} \to K$, такая, что сечение Φ по подмножеству S_{r-1} и разметке $\bar{x}_{r-1}$ имеет непустое ядро. В таком случае существует метка $x_r \in K$, такая, что сечение задачи Φ по подмножеству $S_r = S_{r-1} \cup \{r\}$ и разметке $\bar{x}_r = (\bar{x}_{r-1}, x_r)$ тоже имеет непустое ядро. ■

Сформулируем алгоритм, который для $(\vee,\wedge)$-задачи Φ строит либо последовательность $(\Phi^*_r \mid r = 0,1,\ldots,n)$ задач и последовательность $(x^*_r \mid r = 1,2\ldots,n)$ меток, либо те или иные начальные участки этих последовательностей. Этот алгоритм определен для любой $(\vee,\wedge)$-задачи, а не только для задач, обладающих тем или иным полиморфизмом. Далее будем ссылаться на этот алгоритм как на $(\vee,\wedge)$-алгоритм с самоконтролем.

Алгоритм начинает работу с построения ядра Φ^*_0 исходной задачи Φ. Если ядро Φ^*_0 пустое, то алгоритм прекращает работу и сообщает, что $\bigvee\limits_{\bar{x} \in K^T} \bigwedge\limits_{i,j \in T} \varphi_{ij}(x_i, x_j) = 0$, т.е. решает, что

условия исходной задачи Φ противоречивы. Это решение правильно независимо от наличия или отсутствия какого-либо полиморфизма для решаемой задачи.

Если к началу r-го шага, $r \geq 1$, построена непустая согласованная задача Φ^*_{r-1}, то на r-м шаге выполняется поиск метки $x^*_r \in K$, такой, что сечение задачи Φ^*_{r-1} по объекту r и метке x^*_r имеет непустое ядро Φ^*_r. Если такая метка найдена, то задача Φ^*_r объявляется результатом r-го шага и алгоритм переходит к следующему шагу. Если для исходной задачи Φ существует подходящий полиморфизм, то в силу Леммы 2 искомая метка x^*_r гарантированно будет найдена. Если для задачи Φ такой полиморфизм отсутствует, то искомая метка x^*_r может существовать, а может и не существовать. В последнем случае алгоритм сообщает об отказе от решения задачи. Существенно, что этот отказ возможен только в том случае, когда для исходной задачи отсутствует подходящий полиморфизм.

Если на протяжении всех n шагов алгоритм не сообщил об отказе от решения задачи, то результатом его работы будет разметка $(x^*_r \mid r = 1,2,\ldots,n)$. В силу Леммы 1 для этой разметки выполняется равенство $\bigwedge\limits_{i,j \in S} \varphi_{ij}(x^*_i, x^*_j) = 1$. Это значит, что разметка $(x^*_r \mid r = 1,2,\ldots,n)$ является решением задачи Φ, и этот факт не зависит от наличия или отсутствия подходящего полиморфизма для решаемой задачи.

Таким образом, сформулированный алгоритм определен на множестве всех возможных $(\vee,\wedge)$-задач, которое образует *NP*-полный класс проблем. В области своего определения алгоритм самостоятельно выделяет область своей компетентности – подмножество задач, от решения которых он не отказывается. Область компетентности алгоритма содержит в себе все задачи, для которых существует подходящий полиморфизм. Алгоритм обеспечивает безошибочное решение всех этих и многих других задач, уклоняясь при этом от ответа на трудный вопрос о существовании или отсутствии подходящего полиморфизма для каждой решаемой задачи.

Разрешимые подклассы $(\min,+)$-задач

Субмодулярные $(\min,+)$-задачи. Пусть для каждой пары объектов $i, j \in T$ определена функция φ_{ij} вида $K \times K \to \mathbb{R}$, а не вида $K \times K \to \{0,1\}$, как в $(\vee,\wedge)$-задаче.

Определение 11. Исходными данными $(\min,+)$-задачи является тройка

$$\Phi = \langle T, K, (\varphi_{ij} : K \times K \to \mathbb{R} \mid i, j \in T) \rangle,$$

а собственно задача заключается в вычислении $\min\limits_{\bar{x} \in K^T} \sum\limits_{i,j \in T} \varphi_{ij}(x_i, x_j)$ и $\operatorname{argmin}\limits_{\bar{x} \in K^T} \sum\limits_{i,j \in T} \varphi_{ij}(x_i, x_j)$. ■

В указанном формате можно выразить любую $(\vee,\wedge)$-задачу и поэтому множество $(\min,+)$-задач образует *NP*-трудный класс проблем. Известно, что полиномиально разрешимый подкласс $(\min,+)$-задач образуют так называемые субмодулярные $(\min,+)$-задачи [24]. Определим это понятие в несколько более общем виде, чем общепринятое определение. Пусть для каждого объекта $i \in T$ задана нумерация $\alpha_i : K \to \mathbb{N}$ меток и $A = (\alpha_i \mid i \in T)$ – совокупность этих нумераций.

Определение 12. Задача $\Phi = (\varphi_{ij} : K \times K \to \mathbb{R} \mid i, j \in T)$ называется субмодулярной относительно совокупности $A = (\alpha_i \mid i \in T)$ нумераций, если для любых двух объектов $i, j \in T$ и любых четырех меток $x_i, y_i, x_j, y_j, \in K$, таких, что $\alpha_i(x_i) < \alpha_i(y_i)$ и $\alpha_j(x_j) < \alpha_j(y_j)$, справедливо неравенство $\varphi_{ij}(x_i, y_j) + \varphi_{ij}(y_i, x_j) \geq \varphi_{ij}(x_i, x_j) + \varphi_{ij}(y_i, x_j)$.

Задача Φ называется субмодулярной, если существует совокупность $A = (\alpha_i \mid i \in T)$ нумераций, относительно которой задача субмодулярна. ■

Отметим, что субмодулярность задачи – это достаточно трудно распознаваемое свойство, в то время, как сравнительно просто проверить ее субмодулярность относительно заданной

совокупности нумераций. Общепринятое определение субмодулярности задачи является таким частным случаем Определения 12, когда нумерация α_i не зависит от i, т.е. когда для всех объектов определена одна и та же нумерация.

Существенный прорыв в теории $(\min,+)$-задач и практике распознавания изображений осуществлен в [30–32], где показано, что субмодулярные $(\min,+)$-задачи сводятся к задачам о минимальном сечении графов или, что то же самое, о максимальном потоке в графах. Однако для преобразования конкретной $(\min,+)$-задачи в соответствующую ей задачу о минимальном сечении графа недостаточно гарантировать ее субмодулярность, а необходимо знать еще ту конкретную упорядоченность меток, относительно которой задача субмодулярна. Поэтому этот подход оказался плодотворным прежде всего в задачах стереозрения и сегментации изображений, в которых упорядоченность непосредственно вытекает из прикладного содержания этих задач. Если эта упорядоченность априори не известна, то применение указанного подхода становится проблематичным.

В следующем подразделе описан подход к решению субмодулярных $(\min,+)$-задач, подобный подходу к решению $(\vee,\wedge)$-задач, описанному в подразделе об алгоритме решения таких задач с самоконтролем. Сформулированный далее алгоритм определен на множестве всех возможных $(\min,+)$-задач, а не только субмодулярных. В этой области определения алгоритм самостоятельно обозначает область своей компетентности – подмножество задач, от решения которых он не отказывается. В эту область входят все субмодулярные задачи, и алгоритм решает все эти задачи, минуя проверку субмодулярности каждой из них. Основные идеи описанного далее подхода сформулированы в [16, 17], а затем развиты в многочисленных последующих исследованиях [25–27], в частности, выполненных по Государственной программе «Образный компьютер» [28, 29].

Эквивалентные и тривиальные (min,+)-задачи

Определение 13. Задачи $\Phi = (\varphi_{ij} \mid i, j \in T)$ и $\Phi^* = (\varphi^*_{ij} \mid i, j \in T)$ эквивалентны, если $\sum_{i,j\in T} \varphi_{ij}(x_i, x_j) = \sum_{i,j\in T} \varphi^*_{ij}(x_i, x_j)$ для любой разметки $\bar{x} \in K^T$. ■

Определим число $Pot(\Phi) = \sum_{i,j\in T} \min_{x_i, y_j \in K} \varphi_{ij}(x_i, x_j)$ для каждой $(\min,+)$-задачи Φ и назовем его потенциалом задачи. Очевидно, что

$$\sum_{i,j\in T} \min_{x_i, y_j \in K} \varphi_{ij}(x_i, x_j) \le \min_{\bar{x}\in K^T} \sum_{i,j\in T} \varphi_{ij}(x_i, x_j),$$

однако для некоторых задач это неравенство обращается в равенство.

Определение 14. Задачу $\Phi = (\varphi_{ij} \mid i, j \in T)\rangle$ назовем тривиальной, если

$$\sum_{i,j\in T} \min_{x_i, y_j \in K} \varphi_{ij}(x_i, x_j) = \min_{\bar{x}\in K^T} \sum_{i,j\in T} \varphi_{ij}(x_i, x_j). \quad ■$$

Очевидно, что распознавание тривиальности $(\min,+)$-задачи $\Phi = (\varphi_{ij} \mid i, j \in T)$ сводится к распознаванию непротиворечивости $(\vee,\wedge)$-задачи $\Psi = (\psi_{ij} \mid i, j \in T)$, такой, что $\psi_{ij}(x_i, x_j) = 1$, если $\varphi_{ij}(x_i, x_j) = \min_{u,v\in K} \varphi_{ij}(u, v)$, и $\psi_{ij}(x_i, x_j) = 0$ в противном случае. Полученную $(\vee,\wedge)$-задачу Ψ назовем *бинаризованным представлением* $(\min,+)$-задачи Φ.

Базовая идея описываемого подхода к решению $(\min,+)$-задач состоит в поиске тривиального эквивалента исходной задачи, если такой тривиальный эквивалент вообще существует. Следующие три теоремы указывают реализацию этой идеи.

Теорема 3. Пусть Φ^0 – $(\min,+)$-задача, $Z(\Phi^0)$ – множество $(\min,+)$-задач, эквивалентных Φ^0. Если для задачи Φ^0 существует тривиальный эквивалент, то любая задача Φ^*, такая, что $Pot(\Phi^*) = \max_{\Phi\in Z(\Phi^0)} Pot(\Phi)$ – тривиальна. ■

Теорема доказана в [28]. В этой же работе показано, что поиск задачи Φ^* сводится к максимизации вогнутой функции от $|K|\times|T|^2$ переменных и относится к проблематике выпуклой оптимизации.

Теорема 4. Для любой субмодулярной задачи существует тривиальный эквивалент. ■

Наиболее простое доказательство этой теоремы приведено в работе [33].

Теорема 5. Если $(\min,+)$-задача субмодулярна, то ее бинаризованное представление обладает подходящим полиморфизмом. ■

Доказательство этой теоремы не трудное, но громоздкое и здесь не приводится. Укажем только без доказательства один из возможных полиморфизмов, существование которого утверждает теорема. Коль скоро $(\min,+)$-задача субмодулярная, то существует совокупность $A=(\alpha_i : K\to\mathbb{N}\,|\,i\in T)$ нумераций меток, удовлетворяющая Определению 12 субмодулярности. Искомый полиморфизм $P=(p_i : K\times K\to \to K\,|\,i\in T)$ составлен из операторов p_i, таких, что $p_i(x,y)=x$, если $\alpha_i(x)\ge\alpha_i(x)$, и $p_i(x,y)=y$, если $\alpha_i(x)\le\alpha_i(x)$. Каждый из этих операторов идемпотентен, ассоциативен и коммутативен, т.е. образует полурешетку на множестве K. Указанная совокупность $P=(p_i\,|\,i\in T)$ является полиморфизмом бинаризованного представления любой субмодулярной задачи.

Теоремы 3, 4 и 5 позволяют более определенно сформулировать предложенный подход к решению субмодулярных $(min,+)$-задач на основе их эквивалентных преобразований. Сформулируем алгоритм, который назовем $(\min,+)$-алгоритмом с самоконтролем. Алгоритм состоит из двух этапов. На *первом* этапе входная задача Φ преобразуется в эквивалентную ей задачу Φ^* с максимальным потенциалом. На *втором* – проверяется непротиворечивость $(\vee,\wedge)$-задачи Ψ – бинаризованного представления $(\min,+)$-задачи Φ^* или, что то же самое, проверяется тривиальность задачи Φ^*. Второй этап выполняется $(\vee,\wedge)$-алгоритмом с самоконтролем, описанным ранее. Как было сказано при описании этого алгоритма, он завершает свою работу одним из следующих трех результатов:

- утверждением, что условия задачи Ψ противоречивы;
- утвержденим, что для задачи Ψ отсутствует подходящий полиморфизм;
- формированием разметки $\bar{x}: T\to K$, которая является решением задачи Ψ, а следовательно, и решением задач Φ^* и Φ.

Если входная задача Φ субмодулярна, то завершение работы алгоритма с первым или вторым результатом исключено. Первый результат исключен в силу Теорем 3 и 4, а второй – в силу Теоремы 5. Если же на вход алгоритма поступает произвольная $(\min,+)$-задача, то не исключен ни один из указанных трех результатов, включая и третий. Получение третьего результата обозначает, что найдена тривиальная задача, эквивалентная исходной, и получено решение как этой тривиальной задачи, так и исходной. Из этого не следует, что входная задача Φ субмодулярна. Получение первого или второго результата понимается, как отказ алгоритма от решения текущей задачи, но это возможно только тогда, когда эта задача не субмодулярна.

Приведенный $(\min,+)$-алгоритм с самоконтролем выражает идею решения $(\min,+)$-задач методом их эквивалентного преобразования. Однако он сформулирован не полностью, а с точностью до алгоритма выпуклой оптимизации, к которой сводится поиск задачи с максимальным потенциалом. Для этой цели, как правило, непригодны известные алгоритмы выпуклой оптимизации общего назначения, так как они не гарантируют отыскание оптимума за конечное количество шагов, а осуществляют бесконечную последовательность, предел которой – искомый оптимум. Для реализации изложенного подхода требуется либо разработать специальные конечно-шаговые алгоритмы выпуклой оптимизации, учитывающие специфику $(\min,+)$-задач, либо ослабить требования к решению $(\min,+)$-задач, так, чтобы их мож-

но было удовлетворить известными методами выпуклой оптимизации.

В статье [29] исследован второй из этих двух путей, содержащий три основных момента. Прежде всего, понятие тривиальности обобщено так, что задача $(\varphi_{ij} \mid i,j \in T)$ считается ε-тривиальной, если $\min\limits_{\bar{x}\in K^T} \sum\limits_{i,j\in T} \varphi_{ij}(x_i,x_j) - \sum\limits_{i,j\in T} \min\limits_{x_i,y_j\in K} \varphi_{ij}(x_i,y_j) \le \varepsilon$. Затем разработан алгоритм, который для любого заданного положительного числа $\varepsilon \ge 0$ и для любой субмодулярной задачи (а также ряда других) находит ее ε-тривиальный эквивалент. И наконец, сформулирован алгоритм, который для любого положительного числа δ и любой субмодулярной задачи $(\varphi_{ij} \mid i,j \in T)$ находит разметку $\bar{x}^* \in K^*$, такую, что $\sum\limits_{i,j\in T} \varphi_{ij}(x_i^*,x_j^*) - \min\limits_{\bar{x}\in K^T} \sum\limits_{i,j\in T} \varphi_{ij}(x_i,x_j) \le \delta$. Понятно, что этот алгоритм обеспечивает точное решение $(\min,+)$-задачи, если функции φ_{ij} принимают целочисленные значения.

Поиск заданного количества наилучших решений

В данном разделе исследованы обобщения $(\min,+)$- и $(\vee,\wedge)$-задач, которые будут называться d-оптимизацией и d-совместимостью. Пусть $\varphi: K^T \to \mathbb{R}$ – функция, определенная на множестве разметок. Задача d-оптимизации этой функции, $0 < d <| K |^T$, состоит в поиске подмножества $Sol \subset K^T$, $| Sol |= d$, такого, что $\varphi(\bar{x}) \le \varphi(\bar{y})$ для любых $\bar{x} \in Sol$, $\bar{y} \notin Sol$. Поиск оптимальной разметки в $(\min,+)$-задачах, рассмотренный в предыдущем разделе, является 1-оптимизацией.

Аналогично d-оптимизации, распознавание d-совместимости условий $(\vee,\wedge)$-задачи состоит в ответе на вопрос, существуют ли d различных разметок $\bar{x}_i \in K^T$, $i = 1,2,\ldots,d$, удовлетворяющих условиям этой задачи. Распознавание непротиворечивости условий $(\vee,\wedge)$-задачи, рассмотренной ранее, является распознаванием 1-совместимости.

Необходимость d-оптимизации, $d > 1$, равно как и распознавания d-совместимости, возникает в повседневных, вполне реалистичных ситуациях. Это происходит, например, когда искомый оптимум должен удовлетворять априори известному ограничению, но оптимизация φ при этом ограничении вызывает серьезные вычислительные трудности. В других ситуациях решение x^* предназначено для последующей обработки с помощью алгоритма с ограниченной, но априори неизвестной областью определения. В этом случае можно лишь апостериори констатировать, что текущее полученное решение x^* непригодно для последующей обработки, и запросить иное, тоже хорошее решение. Наконец, формализовать ограничение на решение в принципе невозможно, когда оно адресуется человеку, который может отвергнуть предложенное решение x^* в силу известных только ему соображений. Во всех этих ситуациях целесообразна d-оптимизация с последовательно возрастающим d и отыскание последовательности ухудшающихся решений, пока не будет найдено оптимальное решение, удовлетворяющее объективным, но заранее не известным ограничениям.

Базовый алгоритм *d*-оптимизации. Основополагающая идея d-оптимизации и распознавания d-совместимости принадлежит Лоулеру (*Eugene U. Lawler*) [34] и состоит в следующем. Подобно понятию сечения задачи (см. Определение 10), определим понятие сечения функции $\varphi: K^l \to W$. Функцию $\psi: K^{l-1} \to W$ назовем *сечением* функции φ, если существуют такие $1 \le i \le l$ и $a \in K$, что $\psi(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) = \varphi(x_1,\ldots,x_{i-1},a,x_{i+1},\ldots,x_n)$. Кроме того, определим, что если ψ является сечением φ, а функция ω – сечением ψ, то ω – это тоже сечение φ. Лоулер показывает, что d-оптимизация функции φ сводится к 1-оптимизации определенной совокупности ее сечений, количество которых равно $d \times n \times | K |$. Сформулируем базовый алгоритм d-оптимизации, который в общем случае не обладает преимуществами в

сравнении с алгоритмом Лоулера, но существенно ускоряет поиск d наилучших разметок для некоторых подклассов задач разметки.

Определение 15. Для конечного множества I, упорядоченного множества W, функции $\varphi: I \to W$ и целого числа d, $0 < d <| I |$, выражение $I^* = \underset{i \in I}{\operatorname{argmin}}[d]\varphi(i)$ обозначает, что

$$I^* \subset I, |I^*| = d \ u\ \varphi(i) \leqslant \varphi(j)$$
$$\text{для любых } i \in I^*, j \notin I^*; \qquad (9)$$

для $d \geq | I |$ выражение $I^* = \underset{i \in I}{\operatorname{argmin}}[d]\varphi(i)$ обозначает, что $I^* = I$. ■

Подобно обозначению $\underset{i \in I}{\operatorname{argmin}}\varphi(i)$, которое не обязательно определяет единственный элемент $i^* \in I$, обозначение $\underset{i \in I}{\operatorname{argmin}}[d]\varphi(i)$ не определяет однозначно единственное подмножество $I^* \subset I$. Выражение $I^* = \underset{i \in I}{\operatorname{argmin}}[d]\varphi(i)$ следует понимать как краткую запись утверждения, что подмножество I^* удовлетворяет условиям (9), возможно, наряду с какими-то другими подмножествами.

Будем считать, что упорядоченное множество W содержит элемент, обозначаемый ∞, такой, что $\infty \geq a$ для любого $a \in W$.

Определение 16. Для целого числа d, $0 < d \leq | I |$, выражение $\min_{i \in I}[d]\varphi(i)$ обозначает монотонно неубывающую последовательность значений $\varphi(i)$, $i \in \underset{j \in I}{\arg\min}[d]\varphi(j)$; для $d > | I |$ выражение $\min_{i \in I}[d]\varphi(i)$ обозначает последовательность длины d, которая начинается последовательностью значений $\varphi(i), i \in I$, упорядоченной в неубывающем порядке, и заканчивается $d - | I |$ элементами, равными ∞. ■

В отличие от $\underset{i \in I}{\operatorname{argmin}}[d]\varphi(i)$ последовательность $\min_{i \in I}[d]\varphi(i)$ определена однозначно для любой функции $\varphi: I \to W$.

Пусть $\varphi: K^T \to W$ – функция на множестве разметок, для которой следует построить подмножество $\underset{\bar{x} \in K^T}{\operatorname{argmin}}[d]\varphi(\bar{x})$. Пусть $S \subseteq T$ и $R = T \setminus S$.

Определение 17. Проекцией функции $\varphi: K^T \to \mathbb{R}$ на подмножество S является функция $\varphi_S: K^S \to \mathbb{R}$, такая, что $\varphi_S(x_S) = \min_{x_R \in K^R}\varphi(x_S, x_R)$. ■

Для $S \subseteq T$ введем ради краткости обозначение $Sol_S = \underset{x_S \in K^S}{\operatorname{argmin}}[d]\varphi_S(x_S)$. Построение искомого решения Sol_T основано на следующей взаимосвязи между Sol_S и $Sol_{S \setminus \{i\}}$.

Теорема 6. Пусть $S \subset T$, $i \in S$, $P = S \setminus \{i\}$, $Sol_P = \underset{x_P \in K^P}{\arg\min}[d]\varphi_P(x_P)$.

Пусть $Sol^* \subset K^S$ – некоторое подмножество разметок.

Если $Sol^* = \underset{(x_P, x_i) \in Sol_P \times K}{\arg\min}[d]\varphi_S(x_P, x_i)$, то

$$Sol^* = \underset{x_S \in K^S}{\operatorname{argmin}}[d]\varphi_S(x_S).$$ ■

Теорема доказана в [35].

Из теоремы следует алгоритм *d*-оптимизации, который строит $Sol_{\{i_0\}}$ для подмножества, состоящего из единственного объекта i_0, затем последовательно пересчитывает $Sol_{S \setminus \{i\}}$ в Sol_S и получает в результате $Sol_T = \underset{\bar{x} \in K^T}{\operatorname{argmin}}[d]\varphi(\bar{x})$.

Построение $Sol_{\{i_0\}}$ состоит из следующих вычислений:

- для каждой метки $x_{i_0} \in K$ вычисляют величины $\varphi_{\{i_0\}}(x_{i_0}) = \min_{x_R \in K^R}\varphi(x_{i_0}, x_R)$, где $R = T \setminus \{i_0\}$; сложность этих вычислений равна $C \times | K |$, где C – сложность 1-оптимизации;
- из полученных $| K |$ величин $\varphi_{\{i_0\}}(x_{i_0})$, $x_{i_0} \in K$, выбирают d наименьших; известно [34], что сложность этого выбора линейно зависит от $| K |$, т.е. равна $c \times | K |$, где c – константа. Таким образом, сложность построения $Sol_{\{i_0\}}$ имеет порядок $| K | \times (C + c)$.

Пересчет $Sol_{S\setminus\{i\}}$ в Sol_S состоит в том, что для каждой разметки $x_{S\setminus\{i\}} \in Sol_{S\setminus\{i\}}$ и каждого значения $x_i \in K$ вычисляется величина

$$\varphi_S(x_{S\setminus\{i\}}, x_i) = \min_{x_R \in K^R} \varphi(x_{S\setminus\{i\}}, x_i, x_R),\ R = T \setminus S, \quad (10)$$

а затем из полученных $d \times |K|$ величин выбираются d наименьших. Таким образом, пересчет $Sol_{S\setminus\{i\}}$ в Sol_S имеет сложность порядка $d \times |K| \times (C + c)$, а сложность вычисления $Sol_T = \operatorname{argmin}_{\bar{x} \in K^T}[d]\varphi(\bar{x})$ имеет порядок $d \times |K| \times \times |T| \times (C + c)$.

Алгоритм d-оптимизации, вытекающий из Теоремы 6, так же, как и алгоритм Лоулера, позволяет автоматически решать задачи d-оптимизации для таких подклассов $(\min,+)$-задач, для которых известен алгоритм 1-оптимизации. В частности, это $(\min,+)$-задачи на ациклических структурах и субмодулярные $(\min,+)$-задачи. Кроме того, Теорема 6, так же, как идея Лоулера, позволяет проверять d-совместимость для любого подкласса $(\vee,\wedge)$-задач, для которых известен алгоритм проверки 1-совместимости. Однако из Теоремы 6 следует еще, что для некоторых $(\oplus,\otimes)$-задач d-оптимизацию или проверку d-совместимости можно существенно ускорить. Это те задачи, которые допускают эффективный пересчет проекции φ_S в проекцию $\varphi_{S\setminus\{i\}}$ для любого подмножества $S \subset T$.

***d*-оптимизация в задачах на ациклических структурах**. Пусть $\Phi = \langle T, K, (W, \oplus, \otimes), \mathbb{S}, (\varphi_{ij} \,|\, (i,j) \in \mathbb{S})\rangle$ – условия $(\oplus,\otimes)$-задачи на ациклической структуре $\mathbb{S}$, а ее решение – значение $w^* = \bigoplus_{\bar{x} \in K^T} \bigotimes_{(i,j)\in\mathbb{S}} \varphi_{ij}(x_i, x_j) \in W$. Оптимизационные задачи образуют подкласс этих задач, для которых W – упорядоченное множество, операцией $x \oplus y$ является операция $\min\{x, y\}$, а $\otimes$ – любая операция, образующая с операцией $\min$ полукольцо. Собственно задача заключается в вычислении величины $w^* = \min_{\bar{x} \in K^T} \bigotimes_{(i,j)\in\mathbb{S}} \varphi_{ij}(x_i, x_j)$ – качества наилучшей разметки.

Задача d-оптимизации заключается в вычислении $\min_{\bar{x} \in K^T}[d] \bigotimes_{(i,j)\in\mathbb{S}} \varphi_{ij}(x_i, x_j)$ – качеств d наилучших разметок, и ее можно решать двумя способами. *Первый* способ заключается в применении метода Лоулера или Теоремы 6 для сведения d-оптимизации к решению $(d \cdot |K| \cdot |T|)$ задач 1-оптимизации, каждая из которых решается общим алгоритмом (5, 6) динамического программирования. *Второй*, менее очевидный способ заключается в том, что вычисление $\min_{\bar{x} \in K^T}[d] \bigotimes_{(i,j)\in\mathbb{S}} \varphi_{ij}(x_i, x_j)$ реализуется тем же общим алгоритмом (5, 6), так как вычисление $w^* = \min_{\bar{x} \in K^T}[d] \bigotimes_{(i,j)\in\mathbb{S}} \varphi_{ij}(x_i, x_j)$ является частным случаем общей $(\oplus.\otimes)$-задачи (1, 2). Покажем, что второй способ на порядок более быстродействующий, чем первый.

Теорема 7 [18, 21, 37]. Для коммутативного полукольца $(W, \min, \otimes)$, функций $(\varphi_{ij} : K \times K \to \to W \,|\, ij \in \mathbb{S})$ и числа d существуют коммутативное полукольцо $(W[d], \oplus[d], \otimes[d])$ и функции $(\varphi[d]_{ij} : K \times K \to W[d] \,|\, ij \in \mathbb{S})$, такие, что

$$\bigoplus_{\bar{x} \in K^T}[d] \bigoplus_{ij \in \mathbb{S}}[d] \varphi[d]_{ij}(x_i, x_j) = \min_{\bar{x} \in K^T}[d] \bigoplus_{ij \in \mathbb{S}} \varphi_{ij}(x_i, x_j). \blacksquare$$

Доказательство. Определим $W[d]$ как множество всех монотонно неубывающих последовательностей $(a_1, a_2, \ldots, a_d)$, $a_i \in W$, и для заданного $a \in W$ обозначим $\langle a \rangle \in W[d]$ последовательность, первый элемент которой равен a, а остальные равны ∞.

Пусть $a = (a_1, a_2, \ldots, a_d)$ и $b = (b_1, b_2, \ldots, b_d)$ – две последовательности. Представим совокупность величин $(a_1, a_2, \ldots, a_d, b_1, b_2, \ldots, b_d)$ в виде монотонно неубывающей последовательности и определим сумму $(a \oplus[d] b)$ как последовательность первых d элементов в полученной упорядоченной последовательности.

Представим совокупность величин $a_i \otimes b_j$, $i,j=1,\ldots,d$, в виде монотонно неубывающей последовательности и определим произведение $(a \otimes [d] b)$ как последовательность первых d элементов в полученной упорядоченной последовательности.

Нетрудно убедиться, что коль скоро тройка $(W, \min, \otimes)$ – коммутативное полукольцо с нулевым элементом ∞, то операции $(a \oplus [d] b)$ и $(a \otimes [d] b)$ на множестве $W_{[d]}$ тоже образуют коммутативное полукольцо с нулевым элементом $\langle\infty\rangle$.

Определим функции $(\varphi[d]_{ij} : K \times K \to \to W[d] \mid ij \in \mathbb{S})$ так, что $\varphi[d]_{ij}(x_i, x_j) = = \langle \varphi_{ij}(x_i, x_j) \rangle$. Для любой функции $\varphi : I \to W$ справедливо, что $\bigotimes_{i \in l} [d] \langle \varphi(i) \rangle = \langle \bigotimes_{i \in l} \varphi(i) \rangle$, и поэтому $\bigotimes_{S \in \mathbb{S}} [d] \langle \varphi_S(x_S) \rangle = \langle \bigotimes_{S \in \mathbb{S}} \varphi_S(x_S) \rangle$. Справедливо также, что $\bigoplus_{i \in l} [d] \langle \varphi_i \rangle = \min_{i \in l} [d]\, \varphi(i)$, и поэтому

$$\bigoplus_{x \in K^T} [d] \bigotimes_{S \in \mathbb{S}} [d] \langle \varphi_S(x_S) \rangle = \min_{x \in K^T} [d] \bigotimes_{S \in \mathbb{S}} \varphi_S(x_S).$$

Теорема доказана. ■

Очевидно, что сложность операции $(a \oplus_{[d]} b)$ имеет тот же порядок, что и d-кратное выполнение операции $\min$. Несколько менее очевидно, но все же верно, что сложность операции $a \oplus_{[d]} b$ имеет тот же порядок, что и $(d \log d)$-кратное выполнение операций $\min$ и $\otimes$. Решение $(\oplus, \otimes)$-задачи в произвольном полукольце сводится к $(|K| \times |T|^2)$-кратному выполнению операций $\oplus$ и $\otimes$. В частности, вычисление величины $\min_{\bar{x} \in K^T} \bigotimes_{ij \in \mathbb{S}} \varphi_{ij}(x_i, x_j)$ сводится к $(|K| \times |T|^2)$-кратному выполнению операций $\min$ и $\otimes$, а вычисление $\min_{\bar{x} \in K^T} [d] \bigotimes_{ij \in \mathbb{S}} \varphi_{ij}(x_i, x_j)$ – к $(|K| \times |T|^2)$-кратному выполнению операций $\oplus[d]$ и $\otimes[d]$. Это значит, что d-оптимизация в рассматриваемом классе задач имеет сложность порядка $(d \cdot \log d \cdot C)$, где C – сложность 1-оптимизации, а не $(d \cdot |K| \cdot |T| \cdot C)$, как это было бы при использовании базовых методов d-оптимизации общего назначения.

Покажем в следующих двух подразделах два подкласса $(\min, \otimes)$-задач, для которых d-оптимизацию можно улучшить еще на порядок. Из Теоремы 6 следует алгоритм d-оптимизации, имеющий сложность не $(d \cdot |K| \cdot |T| \cdot C)$ и не $(d \cdot \log d \cdot C)$, а $(c \cdot d \cdot |K| \cdot |T| + C)$, где c на порядок меньше, чем C.

Поиск d наилучших цепочек. Пусть $T = \{1,2,\ldots,n = |T|\}$ и $\mathbb{S} = \{\{r, r-1\} \mid 1 < r \le n\}$. В этом частном случае разметка $\bar{x} : T \to K$ – это цепочка $\bar{x} = (x_1, x_2, \ldots, x_n)$, качество $\varphi(\bar{x})$ которой определено как $\bigotimes_{r=2}^{r=n} \varphi_r(x_{r-1}, x_r)$. Обозначим $\bar{x}_r$ начальный участок разметки $\bar{x}$ длиной r. Будем считать, что при поиске наилучшей цепочки методом динамического программирования исключение объектов посредством оператора (5) происходит в порядке убывания их номеров. Как и прежде, определим семейство вложенных друг в друга подмножеств $S_r \subset T$, $r \in \{1,2,\ldots,n\}$, так что $S_n = T$ и $S_{r-1} = S_r \setminus \{r\}$. Обозначим $\varphi_r^* : K^{S_r} \to \mathbb{R}$ – проекцию целевой функции φ на подмножество S_r и $Sol_r = \operatorname{argmin}_{x \in K^{S_r}} [d] \varphi_r^*(x)$.

Процедура (5) исключения переменной x^r из множества S_r есть не что иное, как построение проекции φ_{r-1}^* на основании уже построенной проекции φ_r^*. Поэтому результатом решения $(\min, \otimes)$-задачи методом динамического программирования является не только величина $\min_{\bar{x} \in K^T} \bigotimes_{ij \in \mathbb{S}} \varphi_{ij}(x_i, x_j)$, но и проекции φ_r^* для всех $r \in \{1,2,\ldots,n-1\}$, что позволяет эффективно строить $\operatorname{argmin}_{\bar{x} \in K^T} [d] \bigotimes_{ij \in \mathbb{S}} \varphi_{ij}(x_i, x_j)$ для любого заданного d.

В соответствии с Теоремой 6 построение $\underset{\bar{x}\in K^T}{\operatorname{argmin}}[d]\bigotimes_{ij\in\mathbb{S}}\varphi_{ij}(x_i,x_j)$ состоит в построении Sol_1 и последовательном пересчете Sol_{r-1} в Sol_r, который заканчивается получением $Sol_n=\underset{\bar{x}\in K^T}{\operatorname{argmin}}[d]\bigotimes_{ij\in\mathbb{S}}\varphi_{ij}(x_i,x_j)$. Построение Sol_1 состоит в вычислении K чисел $\min\limits_{x_R\in K^R}\varphi(x_1,x_R)$, $x_1\in K$, $R=T\setminus\{1\}$, и выборе из них d наименьших. В общем случае для этого требуется решить $|K|$ задач 1-оптимизации. В рассматриваемом случае числа $\min\limits_{x_R\in K^R}\varphi(x_1,x_R)=\varphi_1^*(x_1)$ уже вычислены для всех $x_1\in K$ и поэтому построение Sol_1 состоит лишь в выборе d наименьших чисел из заданных $|K|$ чисел. Сложность этого выбора имеет порядок $|K|$ [34].

Пересчет Sol_{r-1} в Sol_r состоит в вычислении $(d\cdot|K|)$ чисел $\min\limits_{x_R\in K^R}\varphi(\bar{x}_{r-1},x_r,x_R)$, $\bar{x}_{r-1}\in Sol_{r-1}$, $x_r\in K$, где $R=T\setminus S_r$, и выборе из них d наименьших. В общем случае для этого следует решить $(d\cdot|K|)$ задач 1-оптимизации. В рассматриваемом же случае $\min\limits_{x_R\in K^R}\varphi(\bar{x}_{r-1},x_r,x_R)=$ $=\varphi_r^*(\bar{x}_{r-1},x_r)$, а функция φ_r^* построена в процессе исключения объектов и представлена в виде, приемлемом для ее эффективного вычисления. Вычисление ее значения для заданной цепочки $\bar{x}_{r-1}$ и заданной метки x_r сводится к выполнению двух операций $\oplus$. Пересчет Sol_{r-1} в Sol_r в этом случае сводится к вычислению $(d\cdot|K|)$ чисел $\varphi_r^*(\bar{x}_{r-1},x_r)$, сложность которого имеет порядок $(d\cdot|K|)$, и последуюшего выбора из них d наименьших, который имеет ту же сложность $(d\cdot|K|)$. Поиск d наилучших цепочек имеет сложность $(|K|^2\cdot|T|+d\cdot|K|\cdot|T|)$, где первое слагаемое – сложность поиска самой лучшей разметки, а второе – сложность поиска всех остальных. Полученный вывод достаточно неожиданный. Наиболее сложной частью поиска какого-то количества наилучших цепочек является поиск первой, самой лучшей цепочки. Удельные затраты на поиск всех остальных цепочек на порядок меньше.

В работе [21] описан алгоритм поиска d наилучших цепочек за время порядка $(|K|^2\cdot|T|+$ $+d\cdot\log|K|\cdot|T|)$. Величина $(d\cdot\log|K|\cdot|T|)$ – это объем памяти, необходимый для хранения полученных d наилучших цепочек. Это значит, что после того, как найдена самая лучшая цепочка, затраты на поиск всех остальных имеют тот же порядок, что и простое переписывание полученных разметок с одного места памяти в другое. Не исключено, что это оптимальный по сложности алгоритм.

***d*-оптимизация функций, инвариантных относительно мажоритарных операторов.** В работе [23] исследован определенный подкласс $(\min,\max)$-задач $\Phi=\langle T,K,(W,\min,\max),(\varphi_{ij}\,|\,(i,j)\in T)\rangle$. Это задачи, для которых существует совокупность $P=(p_i\,|\,i\in T)$ мажоритарных операторов, таких, что для любых двух объектов $i,j\in T$ и любых шести меток $x_i,x_j,y_i,y_j,z_i,z_j\in K$ выполняется неравенство

$$\max\{\varphi_{ij}(x_i,x_j),\varphi_{ij}(x_i,x_j),\varphi_{ij}(x_i,x_j)\}\geq$$
$$\geq\varphi_{ij}(p_i(x_i,y_i,z_i),p_j(x_j,y_j,z_j)).$$

Собственно задача Ф заключаются в поиске разметки $\bar{x}^*=\underset{\bar{x}\in K^T}{\operatorname{argmin}}[\max\limits_{i,j\in T}\varphi_{ij}(x_i,x_j)]$. Теорема 6 позволяет обобщить этот результат на поиск $\underset{\bar{x}_r\in K^T}{\operatorname{argmin}}[d]\max\limits_{i,j\in T}\varphi_{ij}(x_i,x_j)$.

Как и прежде, будем считать, что $T=\{1,$ $2,\ldots,n=|T|\}$. Для $r\in\{1,2,\ldots,n\}$ обозначим $S_r=\{1,2,\ldots,r\}$, $\bar{x}_r=(x_1,x_2,\ldots,x_r)$, φ_r^* – проекцию функции $\max\limits_{i,j\in T}\varphi_{ij}(x_i,x_j)$ на подмножество S_r и $Sol_r=\underset{\bar{x}_r\in K^{S_r}}{\arg\min}[d]\varphi_r^*(\bar{x}_r)$. Задачу $\Phi=(\varphi_{ij}\,|\,(i,j)\in T)$ назовем согласованной по семейству $(S_r\,|\,r\in T)$, если $\varphi_r^*(\bar{x}_r)=\max\limits_{i,j\in S_r}\varphi_{ij}(x_i,x_j)$ для любого $r\in T$ и любой разметки $\bar{x}_r$.

Представленный в [23] алгоритм решает задачу Φ в два этапа. На *первом* этапе исходная задача $\Phi = (\varphi_{ij} \,|\, (i,j) \in T)$ преобразуется в эквивалентную ей задачу $\Psi = (\psi_{ij} \,|\, (i,j) \in T)$, согласованную по семейству $(S_r \,|\, r \in T)$. Трудоемкость этого этапа имеет порядок $|K|^3 \cdot |T|^3$. На *втором* этапе за время порядка $|K| \cdot |T|^2$ строится разметка $\bar{x}^* = \underset{\bar{x} \in K^T}{\operatorname{argmin}} [\max\limits_{i,j \in T} \psi_{ij}(x_i, x_j)] =$

$= \underset{\bar{x} \in K^T}{\operatorname{argmin}} [\max\limits_{i,j \in T} \varphi_{ij}(x_i, x_j)]$.

Покажем, что если для задачи Φ построен согласованный эквивалент Ψ, то можно находить не только наилучшую разметку за время порядка $|K| \cdot |T|^2$, но и d наилучших разметок за время порядка $d \cdot |K| \cdot |T|^2$. Построение решения $Sol_n = \underset{x \in K^T}{\operatorname{argmin}} [d] \max\limits_{i,j \in T} \varphi_{ij}(x_i, x_j)$ состоит из построения Sol_2 для $S_2 = \{1,2\}$ и последовательности пересчетов Sol_{r-1} в Sol_r до получения решения Sol_n. Поскольку задача Ψ согласована, то построение Sol_2 состоит в выборе d наименьших величин из $|K|^2$ величин $\psi_{12}(x_1, x_2)$, $x_1, x_2 \in K$, и имеет сложность порядка $|K|^2$.

Правило пересчета Sol_{r-1} в Sol_r, которое следует из Теоремы 6, для согласованной задачи Ψ приобретает вид

$$Sol_r = \underset{(\bar{x}_{r-1}, x_r) \in Sol_{r-1} \times K}{\operatorname{argmin}} [d] \max\{\max_{i,j \in S_{r-1}} \psi_{ij}(x_i, x_j), \max_{i \in S_{r-1}} \psi_{ir}(x_i, x_r)\}.$$

Учитывая, что величины $\max\limits_{i,j \in S_{r-1}} \psi_{ij}(x_i, x_j) =$

$= \psi^*_{r-1}(\bar{x}_{r-1})$ уже были получены при формировании Sol_{r-1}, вычисление числа $\max\{\max\limits_{i,j \in S_{r-1}} \varphi_{ij}(x_i, x_j),$ $\max\limits_{i \in S_{r-1}} \varphi_{ir}(x^*_i, k)\}$ для одной пары $\bar{x}_{r-1} \in Sol_{r-1}$, $x_r \in K$ имеет сложность порядка $|T|$, а не $|T|^2$, а сложность вычисления $d \times |K|$ таких чисел – сложность $d \cdot |K| \cdot |T|$. Поиск d наименьших чисел в полученной совокупности $d \times |K|$ чисел имеет сложность порядка $d \times |K|$. Суммарная сложность пересчета Sol_{r-1} в Sol_r имеет порядок $d \times |K| \times |T|$, а $|T|$-кратный пересчет Sol_{r-1} в Sol_r, $r = 3,4,\ldots,n$, завершающийся получением решения

$$Sol_n = \underset{x \in K^T}{\operatorname{arg\,min}} [d] \max_{i,j \in T} \varphi_{ij}(x_i, x_j),$$

имеет сложность порядка $d \times |K| \times |T|^2$.

Таким образом, представленный здесь алгоритм *d*-оптимизации состоит из преобразования решаемой задачи Φ в согласованную задачу Ψ и $|T|$-кратного пересчета Sol_{r-1} в Sol_r. Вычислительная сложность этого алгоритма определяется сложностью первой части, которая не зависит от d и имеет тот же порядок $|K|^3 \times |T|^3$, что и алгоритм 1-оптимизации [23]. Вторая часть алгоритма имеет сложность порядка $d \times |K| \times |T|^2$ и линейно зависит от d с коэффициентом пропорциональности, на порядок меньшим, чем $|K|^3 \times |T|^3$. Этот результат подобен полученному в предыдущем подразделе для поиска d наилучших цепочек. Сложность *d*-оптимизации рассматриваемого класса функций несущественно зависит от *d*. Наиболее сложной частью поиска d наилучших разметок является поиск первой, самой лучшей разметки. Удельные затраты на поиск всех остальных разметок оказываются на порядок меньше.

Заключение. Направление будущих исследований

Представленные результаты исследований проблемы совместимости ограничений формируют одну из парадигм машинного мышления. Основная идея, выражающая отличие машинного мышления от машинного вычисления, состоит в том, что мыслящее устройство решает задачу не на основании заданного алгоритма ее решения, а на основании заданного условия этой задачи. Определение 1 общей $(\oplus,\otimes)$-задач разметки является одной из возможных, но наиболее исследованных реализаций этой идеи. Ей посвящены сотни фундаментальных и прикладных исследований, составляющих мощное направление в современных компьютерных науках. Представленные здесь результаты являются очередным, но далеко не последним этапом исследований указанной проблемы.

Науке о машинном мышлении еще предстоит пройти тот же путь, который прошла наука о машинных вычислениях, но пройти его, по возможности, за более короткое время.

В настоящее время имеется внушительный послужной список важных прикладных задач, решенных благодаря их формулировке в виде $(\oplus,\otimes)$-задачи разметки с последующим применением того или иного известного метода их решения. В частности, это прикладные задачи, решенные в научном коллективе, где выполнена данная работа [8–13]. Эти результаты систематически публикуются, например, в трудах Международной конференции *EMMCVPR* (*Energy Minimization Methods of Computer Vision and Pattern Recognition*). Само название этой конференции говорит, что распознавание образов и, в особенности, изображений составляет обширную область применимости разработанных методов. Опыт разработки таких прикладных программных комплексов показывает, что формулировка условий задачи в формате (1–3) составляет едва ли не самую трудоемкую часть разработки программного комплекса в целом. Формат (1–3) является фактически языком формулировки задачи, подлежащей решению, но это язык самого низкого, можно сказать, атомарного уровня. Сегодняшнее состояние проблематики машинного образного мышления в ее прикладном аспекте аналогично состоянию вычислительной техники многолетней давности, когда было достигнуто понимание, что одна и та же вычислительная машина может выполнить самые разнообразные вычисления, и для этого нужна «всего лишь» программа требуемых вычислений. Однако, как только вычислительные машины стали применять для действительно трудных реальных задач, создание программы вычислений стало изнурительным трудоемким процессом, который нуждался в автоматизации не в меньшей степени, чем само вычисление. Выполнение этого требования проходило через ряд этапов. Поначалу это было составление разнообразных библиотек стандартных подпрограмм, затем – создание языков программирования высокого уровня, средств формального анализа программ, наконец, средств тестирования, отладки и редактирования программ. Строго говоря, потребовалось около четырех–пяти десятилетий, чтобы пройти путь от программирования в кодах операций той или иной вычислительной машины до современных технологий программирования.

Теории и практике образного мышления еще предстоит пройти этот путь. Предположительно, он может состоять из следующих этапов:

- формирования библиотек стандартных структур и весовых функций, наиболее употребимых в распознавании изображений;
- создания языка (языков) высокого уровня, пригодного для формулировки любой $(\oplus,\otimes)$-задачи, но ориентированного на краткую формулировку задач, наиболее уместных в распознавании изображений;
- создания библиотеки известных методов решения $(\oplus,\otimes)$-задач;
- разработки транслятора задачи с языка высокого уровня на язык в формате (1) – (3);
- разработки средств отладки и редактирования задач на языке высокого уровня.

Указанные инструментальные средства должны быть ориентированы на обширный, но возникший только в последние годы круг пользователей. Это не исследователи проблем распознавания и не конечные пользователи той или иной конкретной распознающей системы, а разработчики распознающих систем в самых разнообразных прикладных областях.

1. *Ковалевский В.А.* Оптимальный алгоритм распознавания некоторых последовательностей изображений // Кибернетика. – 1967. – № 4. – С. 75–80.
2. *Ковалевский В.А.* Методы оптимальных решений в распознавании изображений. – М.: Наука, 1976. – 328 с.
3. *Kovalevsky V.A.* Application of Mathematical Programming for Character and Pattern Recognition // Proc. of the 5th Int. Cong. on Cybernetics, Assosiation Int. de Cybernetique, Namur. – 1967. – P. 746–755.
4. *Kovalevsky V.A.* Recognition by Imitating the Process of Pattern Generation / Watanabe S. (Ed): Methodologies of Pattern Recognition. – Acad. Press. – 1969. – P. 345–358.
5. *Винцюк Т.К.* Распознавание устной речи методами динамического программирования // Кибернетика. – 1968. – № 1. – С. 81–88.

6. *Винцюк Т.К.* Анализ, распознавание и интерпретация речевых сигналов. – Киев: Наук. думка, 1987. – 262 с.
7. *Rossi F.*, *van Beek P.*, *Walsh T.* Handbook of Constraint Programming, Foundations of Artificial Intelligence. – Elsevier, 975 p.
8. *Рябоконь Д.И.* Пространственная реконструкция поверхностей по стереопаре изображений с помощью алгоритма поиска минимального сечения на графе // УСиМ. – 2004. – № 3. – С. 47–51.
9. *Рябоконь Д.І.* Технологія побудови тривимірних моделей неперервних поверхонь за стереопарами зображень: Дис. … канд. техн. наук. – К.: МННЦ ІТіС, 2005. – 138 с.
10. *Ковтун И.В.* Технология текстурной сегментации изображений на основании марковских случайных полей и решения (max,+)-задач // УСиМ. – 2004. – № 2. – С. 61–66.
11. *Ковтун І.В.* Сегментація зображень на основі достатніх умов оптимальності в *NP*-повних класах задач структурної розмітки: Дис. … канд. техн. наук. – К.: МННЦ ІТіС, 2004. – 135 с.
12. *Tyshchenko M.A.* 3D reconstruction of human face based on single or several images // УСиМ. – 2011. – № 2. – С. 79–85.
13. *Тищенко М.А.* Тривимірна реконструкція людського обличчя в задачах ідентифікації особи: Дис. … канд. техн. наук. – К.: МННЦ ІТіС, 2011. – 118 с.
14. *Cooper M.C.* Reduction operations in fuzzy or valued constraint satisfaction, Fuzzy Sets and Systems 134. – 2003. – P. 311–342. – www.elsevier.com/locate/fss
15. *The* complexity of soft constraint satisfaction / D.A. Cohen, M.C. Cooper, P.G. Jeavons et al. // Artificial Intelligence 170. – 2006. – P. 983–1016. – www.elsevier.com/locate/artint
16. *Шлезингер М.И.* Синтаксический анализ двумерных зрительных сигналов в условиях помех // Кибернетика. – 1976. – № 4. – С. 113–130.
17. *Коваль В.К.*, *Шлезингер М.И.* Двумерное программирование в задачах анализа изображений // Автоматика и телемеханика. – 1976. – № 8. – С. 149–168.
18. *Шлезингер М.И.* Математические средства обработки изображений. – Киев: Наук. думка, 1989. – 197 с.
19. *Flerova N.*, *Marinescu R.*, *Dechter R.* Searching for the M Best Solutions in Graphical Models // J. of Artificial Intelligence Research. – 2016. – N 55. – P. 889–952.
20. *Ruttkay Zs.* Fuzzy constraint satisfaction // Proc. 3rd IEEE Int. Conf. on Fuzzy Syst. – 1994. – P. 1263–1268.
21. *Шлезингер М.*, *Главач В.* Десять лекций по статистическому и структурному распознаванию. – К.: Наук. думка, 2004. – 545 с.
22. *Cohen D.A.*, *Jeavons P.G.* The Complexity of Constraint Languages, Chapter 8, in Handbook of Constraint Programming. – Elsevier, 2006. – P. 245–280.
23. *Водолазский Е.*, *Флах Б.*, *Шлезингер М.* Минимаксные задачи дискретной оптимизации, инвариантные относительно мажоритарных операторов // ЖВММФ/ – 2014. – Т. 54, № 8. – С. 1368–1378.
24. *Lovasz L.* Submodular functions and convexity. Ed. by *A. Bachem*, *M. Grotschel*, *B. Korte* // Mathematical Programming – The State of the Art. – Springer-Verlag, New York, 1983. – P. 235–257.
25. *Schlesinger M.*, *Flach B.* Some solvable subclasses of structural recognition problems, Czech Pattern Recognition Workshop, 2000. – P. 55–62.
26. *Werner T.* A Linear Programming Approach to Max-sum Problem: A Review // IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI). – July 2007. – **29**(7). – P.1165–1179.
27. *Werner T.* Revisiting the Linear Programming Relaxation Approach to Gibbs Energy Minimization and Weighted Constraint Satisfaction // Ibid. Aug. 2010. – **32**(8). – P. 1474–1488.
28. *Шлезингер М.И.*, *Гигиняк В.В.* Решение (max,+)-задач структурного распознавания с помощью их эквивалентных преобразований // УСиМ. – 2007, Ч. 1, № 1, С. 3–15, Ч. 2, № 2, С. 5–17.
29. *Шлезингер М.И.*, *Антонюк К.В.* Анализ алгоритмов диффузии для решения оптимизационных задач структурного распознавания // Кибернетика и системный анализ. – 2011. – № 2. – С. 3–12.
30. *Ishikawa H.*, *Geiger D.* Segmentation by grouping junctions // IEEE Comp. Society Conf. on Computer Vision and Pattern Recognition. – 1998. – P. 125–131.
31. *Boykov Y.*, *Veksler O.*, *Zabih R.* Fast Approximate Energy Minimization via Graph Cuts. IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI). – 2001. – 23(11) – P. 1222–1239.
32. *Kolmogorov V.*, *Zabih R.* What energy functions can be minimized via graph cuts? / Eur. Conf. on Comp. Vision (ECCV). – Springer-Verlag, 2002. – P. 65–81.
33. *Шлезингер М.*, *Вернер Т.* О супермодулярной максимизации. – http://www.irtc.org.ua/image/app/webroot/Files/publications/Schlesinger/Smachivanije.pdf
34. *Lawler E.L.* A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and Its Application to the Shortest Path Problem, Management Science. Theory Series. – Mar. 1972. – **18**, N 7. –P. 401–405.
35. *Schlesinger M.*, *Flach B.*, *Vodolazskiy E.* M-best solutions for a class of fuzzy constraint satisfaction problems (Submitted on 23 Jul 2014), arXiv:1407.6166v1.
36. *Time* bounds for selection / M. Blum, R. Floyd, V. Pratt et al. // J. of Comp. and Syst. Sci., 1973. – **7**, N 4. – P. 448–461
37. *Rollon E.*, *Flerova N.*, *Dechter R.* Inference Schemes for M Best Solutions for Soft CSPs Article // Proc. of Workshop on Preferences and Soft Constraints. – 2011. – P. 124–138.

Поступила 27.03.2017
Тел. для справок: +38 044 526-6208 (Киев)
E-mail: *schles@irtc.org.ua*

UDC 519.683:681.3

M.I. Schlesinger

Pattern Recognition as an Implementation of a Certain Type of Thought Processes

The paper presents a review of pattern recognition research conducted by the International Research and Training Center for Information Technologies and Systems during the last 20 years since the moment of it's foundation. This research lies on the edge of classical pattern recognition and constraint satisfaction problems. The system of concepts, problems and algorithms produced by the merge of these fields formalizes a particular type of thought process performed by humans and other living beings.

The application of Constraint Satisfaction Problem theory to pattern recognition problems has produced a breakthrough in such traditionally hard problems in computer vision as stereo vision and texture segmentation. At the same time, the merge of Constraint Satisfaction Problem theory and practical computer vision problems has led to expansion of mathematical theory of the former. First of all it has resulted in introduction of a quality function over the set of solutions and finding the best solution instead of an arbitrary one. The next generalization consisted in finding a given number of best solutions and not just a singe best solution.

The paper describes methods of finding the best solution to the Weighted (Soft) Constraint Satisfaction Problem as well as the method of finding any given number of best solutions. These methods are implemented as algorithms whose domain is the set of all possible Weighted (Soft) Constraint Satisfaction Problems, i.e. a NP-hard problem class. For any given problem from the domain the algorithms either find it's solution or reject the problem. It is essential that the algorithms automatically distinguish the subdomains of their competence, i.e. the subset of problems that they do not reject. The subdomain of competence of the algorithm that finds the best solution includes the known class of submodular minimization problems but is not restricted to it. The subdomain of competence of the algorithm that finds a given number of best solutions includes the minimization of functions with a majority polymorphism but is not restricted to it.