УДК 004.8

Д. Зербіно, Ю. Цимбал, Ю. Кинаш
Національний університет "Львівська політехніка",
кафедра автоматизованих систем управління

САМООРГАНІЗАЦІЯ В МОДЕЛЯХ ОБЧИСЛЕНЬ НЕЙРОННИХ МЕРЕЖ І ДЕКЛАРАТИВНИХ ПРОГРАМ

Порівнюються дві моделі обчислень – штучні нейронні мережі та декларативні програми, що побудовані на основі логіки предикатів. Пропонуються такі їх узагальнення, за яких процес обчислень зможе приводити до цілеспрямованих змін власної програми. Сформульовано принципи самоорганізації, за якими вказані зміни будуть не хаотичними, а визначеними в результаті пошуку. Для детального вивчення самоорганізації необхідне зближення та взаємне доповнення розглянутих моделей.

Ключові слова: PROLOG, штучні нейронні мережі, модель обчислень, самоорганізація, генетичні алгоритми.

The two models of computations have been compared – the artificial neural networks and declarative programs based on logic. The generalization of these models is proposed in the part where the computational process can cause changes of the program. The principles of self-organization for these models have been formulated. According to these principles the above-mentioned changes will be determined as a result of the search. Complementarity of the discussed models is necessary for a profound study of the self-organization.

Key words: PROLOG, artificial neural networks, model of computations, self-organization, genetic algorithms.

Вступ

Загальноприйнято, що живим організмом є високоорганізована система, яка цілеспрямовано підтримує своє існування в змінному середовищі і може відтворювати подібні до себе системи з позитивними змінами, які покращують властивості пристосування до середовища. Це визначення цілком підходить до терміна "самоорганізація", який додатково підкреслює, що в результаті впорядкованість системи загалом зростає і причиною цього є сама система.

Якщо самоорганізація справді існує в природі, то "теоретично живими" можуть виявитись і квантові системи на поверхні Сонця, і кристалічні структури на дальніх планетах за дуже низьких температур, і штучно створена технічна система. Для того, щоб ця стаття не була науково-фантастичною, необхідно з'ясувати:

- чи існують якісь загальні принципи самоорганізації, без яких вона неможлива;
- якщо принципи самоорганізації існують, то чи можливо їх застосувати до програмних систем у вигляді формальних операторів;
- чи можливо на цих принципах започаткувати нову технологію програмування, яка б давала змогу програмним способом фіксувати всі позитивні зміни в програмах з метою їх самовдосконалення, а також автоматично впорядковувати цілі програми та оптимізовувати способи їх досягнення та модифікувати алгоритм загалом.

Загальновідомо, що для ефективного управління системою її складність не повинна бути високою, інакше керувати усіма процесами в ній стане неможливим. Отже, високоорганізована система не може існувати без законів саморегулювання та самоорганізації, які повинні спрощувати управління за рахунок виникнення спонтанних контурів регулювання. Коли живі організми досягають певного рівня організації, вони починають взаємодіяти й утворювати інші системи зі своїми контурами регулювання, що також є проявом самоорганізації. Феномен виникнення багатогранних форм життя дотепер не має пояснення, що вказує на актуальність дослідження алгоритмів самоорганізації. Ми також вважаємо, що потрібно дослідити, як утворюються

високоорганізовані системи і як в них виникають спонтанні контури регулювання, що координують загальне управління системою.

Метою цієї роботи є лише перший крок – уточнення понять адаптації та самоорганізації, виявлення ознак та принципів самоорганізації, а також спроба виявити в моделях штучних нейронних мереж та декларативних програм властивості, які сприяють появі ознак самоорганізації.

Основні принципи самоорганізації

В навколишньому середовищі існують складні, але не самоорганізовані системи. Самоорганізація є необхідною, але недостатньою властивістю живої системи. Розглядатимемо самоорганізацію як цілеспрямовану зміну власної структури (можливо, для оптимізації контурів регулювання в ній). Отже, щоб зробити цілеспрямований крок, необхідно визначити можливі альтернативні кроки і після цього з'ясувати, яка з альтернатив краща і за якими критеріями. Частковими випадками є такі: всі альтернативи перелічити неможливо; критеріїв занадто багато. В такому випадку шукають проблему і намагаються її вирішити через аналіз логічних зв'язків. Більшість проблем взаємозв'язані і, на відміну від адаптації, системи з принципами самоорганізації вирішують проблеми у комплексі.

Очевидно, що "вгадати" правильне рішення для системи просто неможливо. Тому логічно припустити, що в самоорганізованих системах існує прихований алгоритм перебору варіантів з логічним аналізом кожного з них, після якого система вибирає найкращий. Властивість самоорганізованості системи не залежить від її природи. Наприклад, в [1] розглянуто прості правила заміни даних на дискретній площині, які приводили до цілеспрямованого розмноження певних конфігурацій даних.

У простих системах всі закони функціонування визначає зовнішнє середовище. Під їх впливом "відсіюються" початкові конфігурації даних, у результаті якого одні конфігурації зберігаються і навіть розмножуються, а інші деградують, переходять в іншу форму і "відмирають". А у складних системах існує деякий суб'єкт, що сам вирішує, які дані "показати" назовні, щоб згідно з законами середовища досягти певної мети. Тому пропонуємо одразу ж розділити самоорганізацію на спонтанну (яка залежить лише від законів середовища) і самоорганізацію цілеспрямовану (за наявності керуючого суб'єкта). Якщо керуючий суб'єкт має занадто примітивні функції, то самоорганізація може бути лише спонтанною, тобто в основному залежати від законів середовища. Чим більше можливостей у суб'єкта "приховувати" від середовища дані, тим "розумнішою" і незалежнішою може бути самоорганізація.

Для вивчення цього феномену спочатку розглянемо штучні нейронні мережі, які розробники задумували як альтернативу до класичних процесорів і які відрізнялися за принципом обчислень. Якщо процесори класичної архітектури вимагають від програміста написання програми, основаної на моделі задачі, то обчислення в нейромережах, на нашу думку, повинні ґрунтуватись на таких принципах:

- **нейромережа сама створює модель об'єкта, яким керує.**

Цей принцип назвемо першим принципом самоорганізації нейромережі.

Наприклад, необхідно створити штучну нейронну мережу для розпізнавання слова "самоорганізація", що зображується у формі оцифрованого сигналу з певною точністю. В цій задачі розрізнятимемо два підходи: параметричний підхід та алгоритмічний. За параметричного підходу просто змінюватимемо масштабування сигналу так, щоб отриманий сигнал максимально збігався з взірцем. Масштабування може бути нелінійним. По горизонталі задаємо частотне масштабування, а по вертикалі – амплітудне. Зрозуміло, що такий підхід не дасть якісних результатів, якщо диктор затягує чи скорочує букви або говорить стурбована людина з іншим тембром голосу. Спробу подолання цих проблем зроблено в [2].

Алгоритмічний підхід ґрунтується на виділенні певних логічних ознак сигналу (за принципом багатократного аналізу – з отриманням первинних, вторинних, третинних ознак тощо). Сама нейромережа повинна вибрати, які з цих ознак є суттєвими, а які – ні. Наприклад, для розпізнавання звуку "Р" істотним є процес коливання язика, який можна розпізнати або візуально, або через характерний періодичний звук з різкими коливаннями на першому фронті. Зверніть увагу, що частота

коливань для розпізнавання звуку є менш інформативною, ніж спроба диктора поворушити язиком в комбінації з гортанню саме так, щоб вийшов звук “Р”. Під час вимови довільних звуків інформацію дає або послідовність рухів, або сила напруження м’язів відносно деякого середнього рівня. У будь-якому випадку за характером звуку слухач внутрішньо інтерпретує, як рухаються м’язи легень, гортані, язика і губ, і завдяки цьому розуміє, що хотів сказати диктор, навіть якщо звук спотворений.

У наведеному вище прикладі суттєвими є алгоритми виділення ознак. Їх кількість відображає різноманітність та глибину аналізу. Отже, алгоритмічний підхід у розпізнаванні ґрунтується на алгоритмах, що представляють логіку виділення ознак на основі інтерпретації кожної найменшої зміни в сигналі. Людина інтерпретує сигнал за ознаками бажання диктора вимовити той чи інший звук, а саме за передаванням цього бажання через зміни напруження різних груп м’язів.

Саме інтерпретація сигналу як носія інформації можлива лише у тому випадку, якщо у “розпізнавача” існує певна модель процесу і якщо він розуміє за суперечливими зовнішніми ознаками, як саме відбувається процес, а також які події в ньому стаються. Наприклад, інтерпретуючи найдрібніші ознаки у зображенні обличчя, людина може моментально розпізнати настрій та емоції іншої людини, але настрій жуків чи птахів розпізнати важко, оскільки модель розпізнавача в цьому випадку є неповною.

Звичайно, що нейромережа не є суперінтелектом, який уявляє всі деталі вимови, але вона може створити модель вимови необхідних звуків на якомусь примітивному рівні, якого достатньо для розпізнавання не лише слів, але і емоцій диктора. Отже, перший обов’язковий крок до самоорганізації в розпізнаванні – це побудова хоча б і примітивної, але несуперечливої моделі, в якій відбувається розпізнавання. В процесі самоорганізації ця модель буде вдосконалюватися.

Щоб нейронна мережа прийняла правильне рішення, розглянутого вище “принципу побудови моделі” виявляється недостатньо. Необхідно також встановити мету обчислень, тобто мету прийняття рішення (або мету розпізнавання), а також критерії оцінювання, які відіграють істотну роль в зворотних зв’язках системи. Отже, сформулюємо другий принцип самоорганізації для нейронних мереж, в який входить мета:

- **в процесі обчислень нейромережа сама уточнює мету обчислень та вдосконалює критерії оцінювання.**

Ця мета може бути ієрархічною, багатокомпонентною та невпорядкованою. Під час самоорганізації головну роль відіграє системоутворювальний фактор, тобто мета, з якою елементи утворили систему або взаємодіють між собою. Зміна системоутворювального фактора веде самоорганізовану систему до краху.

Завдання критеріїв оцінювання можна розглядати як частковий порядок на множині ознак самого обчислювального процесу. Це означає, що для самоорганізації система повинна контролювати сама себе і визначати ознаки та параметри своєї роботи. Частковий порядок на множині ознак визначає, які з ознак бажані, а які – небажані. Ознаки в загальному випадку являють собою числа різних специфічних типів, наприклад, ймовірність, тривалість, періодичність, відносний рівень, кількість тощо. Ознаки утворюються не хаотично, а у зв’язку з наближенням чи віддаленням мети обчислень. Критерії того, наскільки близько досягнута мета, формулюються окремо. Зазвичай порівнювати можна лише ознаки однакового типу, і на основі цих порівнянь самоорганізована система може вибирати кращий варіант функціонування.

Алгоритм обчислень складається з арифметико-логічних операцій, операцій порівняння та операцій за умовою. Визначаючи результати операцій, необхідно уникати такої звичної дії, як присвоєння. Саме через присвоєння виникають труднощі при модифікації алгоритмів у системах. Тому для самоорганізованих систем всі результати обчислень повинні бути тимчасовими, тобто такими, щоб за необхідності можна було б без проблем їх замінити, не переробляючи інших даних, з ними зв’язаних. Це можна реалізувати спеціальною ієрархічною моделлю пам’яті, в якій всі дані мають статус припущень.

У сучасному нейромережевому програмуванні обидва принципи самоорганізації реалізуються як навчання, в результаті якого на конкретних прикладах визначається модель об’єкта, а також мета обчислень. Фактично, в результаті навчання уточнюються коефіцієнти зв’язків між нейронами сусідніх шарів або коефіцієнти зворотних зв’язків (залежно від топології нейронної мережі).

Приклад створення самоорганізованої системи

Уявімо, що нам необхідно створити нейронну мережу для розпізнавання слова, яке вимовляють різні люди. Розв'язання задачі безпосередньо за допомогою "звукового персептрона" [3] не дає очікуваного ефекту, тому що це привело би до необхідності масштабувати слово до фіксованої кількості рецепторів, не меншої, ніж кількість поділок оцифрованого сигналу, наприклад, 8000 (оскільки навчання повинно вестися для фіксованої кількості нейронів). По-друге, довелося б виділяти окремі слова, що інколи достатньо складно.

Пропонуємо будувати системи з елементами самоорганізації не "просто напряму", а спираючись на деякі базові знання. Так, для розпізнавання звуків, що видає людина, необхідно накопичити базові знання: а) створити базу зразків звуків, що утворює людина при фіксованому подиху, фіксованому отворі рота і різному напруженні певних м'язів гортані; б) створити базу зразків звуків при розслабленій гортані, фіксованому подиху, фіксованому отворі рота і різному положенні язика; в) створити інші бази звуків, що утворює людина, коли всі параметри фіксовані, а один змінюється.

Тепер процес самоорганізації зведено до того, щоб виділити параметри сигналу, які однозначно характеризують напруження досліджених груп м'язів, з метою добитися, щоб система безпомилково визначала групу напружених м'язів, подих легень, положення язика і рота під час вимовленої диктором фонеми. На цьому завершується побудова моделі вимову, що спирається на так звані "базові знання". Розпізнавання слів можна розпочинати лише після успішного розв'язання вказаної задачі побудови моделі вимови, представивши кожне слово, що вимовляє диктор, через послідовність або комбінацію напружень певних груп м'язів, що входять у модель. Поки що ці властивості нейромереж реалізовано на порівняно примітивних задачах і моделях, які можна порівняти з "умовними рефлексами" на певні структури даних. Для подальшого розвитку самоорганізованих систем необхідно зробити декілька очевидних кроків.

1. Для пошуку найкращого оптимального рішення в самоорганізованих системах повинен бути передбачений механізм повного перебору всіх можливих варіантів розв'язання поставленої задачі з можливістю продовження цього перебору з довільного місця.
2. Для скорочення механізмів повного перебору в самоорганізованих системах має бути механізм логічних тверджень, що розділяє варіанти на більш перспективні і менш перспективні.
3. У процесі пошуку розв'язку задачі система може висувати логічні твердження на основі базових знань про задачу, які надалі можуть використовуватися в алгоритмах виділення ознак.
4. Принцип накопичення досвіду, що залежить від кількості та різноманітності розв'язаних задач, реалізується у накопиченні висунутих системою логічних тверджень.
5. У самоорганізованих системах необхідно перевести логіку на універсальний рівень, використовуючи принцип узагальнення та підстановки понять. Він полягає у тому, що кожний алгоритм може бути елементом іншого, батьківського алгоритму, а також цей самий алгоритм може виступати в ролі батьківського алгоритму. Тобто алгоритми повинні бути універсальними елементами формально-логічної системи, які можна заміняти.

Поки що не можна сказати, що сучасні нейромережі готові реалізовувати усі ці властивості, способи їх навчання є недосконалими, оскільки не дають змоги робити логічні припущення. В сучасних нейромережах логічний аналіз насправді заміняється на ймовірність, яка не може передати певну умовність понять, що логічно залежать від ситуації.

Декларативний підхід до опису самоорганізації

Одночасно з технологіями нейромереж розвивається такий напрям, як декларативні програми, який раніше вважався недосконалим з погляду практичного програмування. Основною причиною такої "нелюбові" розробників програмного забезпечення є непередбачуваність поведінки декларативної програми, оскільки програміст задає не сам алгоритм, а систему логічних тверджень, на яких алгоритм побудує виконавча система. Очевидну з погляду алгоритмічної мови програмування задачу декларативною мовою треба формулювати зовсім неприродним способом. Звичайно, що при логічній інтерпретації такої декларативної програми виникають різноманітні "побічні ефекти", що потребують додаткових логічних уточнень, які дратують програмістів. Крім того, транслятор все одно перетворює декларативну програму на алгоритмічну.

З розвитком нейромережевих технологій декларативне програмування набуває нового змісту. На нашу думку, його можна розвивати і в напрямі самоорганізаційних технологій.

Основною особливістю системи, яка здатна до самоорганізації, є те, що в процесі самоорганізації вона не руйнується, а навпаки, стає стійкішою та надійнішою. Якщо в програмі, що створена на будь-якій алгоритмічній мові, спробувати щось поміняти, то це, швидше за все, приведе до краху всього програмного продукту, чого не стається з декларативною програмою. Отже, в декларативному підході можливі специфічні мови та методи програмування, які покращують власний програмний код.

Мова програмування – це універсальний спосіб спілкування з обчислювальною системою, незважаючи на спосіб обчислень, покладений в основу її функціонування. Одним з істотних недоліків сучасних мов програмування високого рівня є неможливість природно та логічно обґрунтовано модифікувати власну програму. Тобто в сучасних мовах програмування не існує засобів, які б представляли алгоритм програми як дані з можливістю його коректної модифікації. Зроблено лише один невеликий крок в цьому напрямку – це генетичні алгоритми [4]. У генетичному програмуванні дозволені два типи модифікації програм, основою яких є випадковість – схрещування та мутація. Як показали експерименти над реальними програмами, які цим способом синтезують прості логічні формули, суттєві труднощі виникають вже з формулами, що містять п’ять змінних. Вони виходять або наближеними, або дуже складними. З цього можна зробити висновок, що повністю випадковим способом програма з елементами самоорганізації діяти не повинна.

Зауважте, що сучасні процесори теоретично мають можливість модифікування власної програми – для цього необхідно лише результати роботи програми спрямовувати в сегмент пам’яті програмного коду. Інакше кажучи, технічні можливості сучасних процесорів дають змогу зробити методи та засоби створення самоорганізованих програм та мов програмування з елементами самоорганізації.

Із загальних міркувань можна припустити, що програму, яка здатна до самоорганізації, повинна попередньо написати людина-програміст, але програмний код сам може вносити в себе зміни в деяких межах. Наприклад, розглянуті вище програми для синтезу функції на основі генетичних алгоритмів виконують перебір всіх можливих комбінацій базових функцій, з яких будується синтезована функція. Логіка скорочення перебору основана на припущенні, що синтезована функція не дуже сильно відрізняється від базової.

Отже, у програмі, здатній до самоорганізації, необхідно чітко описати, які зв’язки є базовими, а які можна модифікувати, і які правила не можна порушувати під час модифікації.

Методи коректної модифікації програмного коду

Отже, для реалізації напряму самоорганізації необхідно передусім ввести в мову програмування засоби, що дають змогу модифікувати власний код програми так, щоб не привести алгоритм до краху. Найзручніше це зробити в декларативних мовах програмування типу PROLOG. Оскільки декларативна мова складається з предикатів, які виражають деякі зв’язки між поняттями, що програмуються, то для програм, які змінюють власний код, повинні бути визначені спеціалізовані предикати для зміни предикатів. Це добре узгоджується з концепцією, що початкову програму створює програміст, а алгоритм модифікується в процесі її виконання введенням у програму нових, досконаліших предикатів. Для реалізації такої концепції необхідно:

1. Відмовитись у мові програмування від будь-яких обмежень у оперативній пам’яті: за кількістю змінних, альтернатив, на довжину предикатів, змінних, списків тощо, тому що будь-яка прив’язка до конкретних значень веде до невдачі під час модифікації.
2. Ввести спеціалізований предикат COMPILE(“НАЗВА ФАЙЛА(X_1,X_2,…,X_N)”, ERR), який зможе виконати компіляцію заданого текстового файла. Якщо компіляція відбулася успішно і код помилки ERR=0, то викликається найперший предикат зі щойно скомпільованого файла з параметрами (X_1,X_2,…,X_N). Через ці ж змінні в основну програму повертається результат у разі успіху предиката. У випадку зациклювання предиката (що веде до переповнення стека) повертається код ERR=1.
3. Навчити програму узагальненню, в основу якого покладена заміна нескінченних послідовностей скінченними позначеннями, а також дозволити позначення будь-яких даних як другий, третій і т.д. шар нових даних.

4. Ввести операції додавання предиката в розділ “Proposals”, додавання атрибута в предикат, а також операцію видалення предиката.
5. Для полегшення програмування ввести спеціалізовані предикати перебору, наприклад, перебору всіх підмножин чи підрядків, що задовольняють задану умову.
6. Для розв’язання суперечливих задач ввести операцію “...більш бажана, ніж...”.

Особливості самоорганізованих програм

Розглянемо схему класичної декларативної програми без самоорганізації:

Множина можливих варіантів → Правило, яке описує мету → Перебір варіантів, які задовольняють правило → Вибір найкращого варіанта.

Схема програми з елементами самоорганізації:

Правило, що описує мету, можливі дії та початковий стан → Аналіз досягнення мети → Формування понять, що віддаляють або наближають мету → Накопичення перспективних варіантів (гіпотез) → Перебір варіантів (розгляд гіпотез) → Вибір найкращого варіанта.

В обох програмах є правило, яке описує мету, перебір варіантів та вибір найкращого варіанта. Але в першій програмі множина варіантів, з якої необхідно вибрати найкращий, задається апріорно, тобто перед виконанням алгоритму розв’язку. Вхідними даними програми є лише початковий стан задачі. В другій програмі також апріорно задаються деякі варіанти дій, але їх зміст зовсім інший – ці варіанти розглядаються як обмеження на обчислення, які не можна порушувати в процесі пошуку рішення. Тому правило, що описує мету, та правила, що задають певні обмеження, виступають як одне ціле.

Самоорганізація зумовлює, що програма не просто вибирає елемент із запропонованої множини, а може цілеспрямовано підбирати композицію із запропонованих елементів. За такого підходу необхідний потужний логічний блок аналізу досягнення мети. В результаті аналізу мети програма повинна знайти множину понять, які наближають або віддаляють мету.

Що таке при тому поняття? Поняттям є деяка множина, яка задається правилом. Отже, оперуючи логічними умовами, насправді ми оперуємо множинами, які їх представляють. Отже, вибір найкращого варіанта – це пошук максимального елемента з перетину множин, які задають обмеження на цю задачу.

Однак не кожне правило задає поняття. Швидше з поняттям пов’язаний алгоритм, що дає деяку кількісну характеристику. Наприклад, якщо розглядати шахові задачі, то поняттями можуть бути:

“*цінність фігури*” – за кількістю клітин, на які вона може перейти (виняток – король);

“*атака*” – коли фігура меншої цінності нападає на фігуру більшої цінності;

“*вилка*” – фігура одночасно атакує більше від однієї фігури суперника;

“*зв’язка*” – фігура не може рухатись, оскільки відкриває можливість атаки;

“*цінність позиції*” – за кількістю активних фігур, а також за кількістю можливих атак з нашої сторони і неможливістю атак суперника;

“*цінність ходу*” – цінність позиції після оптимальної відповіді суперника на цей хід.

Реалізація логіки самоорганізації

Основою аналізу досягнення мети повинен бути механізм заміни кожного рівняння $\Psi_i(X,Y,\ldots, Z)$, що описує задачу, множинами його розв’язків $\{(x_1,y_1,\ldots,z_1), (x_2,y_2,\ldots,z_2), \ldots\}$, де $(x_i,y_i,\ldots,z_i)$ – це окремий розв’язок, який задовольняє рівняння $\Psi_i(X,Y,\ldots, Z)$. Система рівнянь означатиме логічну кон’юнкцію правильності кожного рівняння, тобто перетин множин розв’язків кожного рівняння. Здебільшого скінченні рівняння необхідно замінити нескінченними множинами їх розв’язків. На перших кроках для представлення нескінченних множин можна обмежитись деякими n-скінченними множинами рішень цих рівнянь в поєднанні з розв’язками, що видають алгоритми Ω_i, які при циклічному застосуванні дадуть змогу отримати решту розв’язків $\{\Omega\}$ з нескінченної множини для *i*-го рівняння (окрім тих, які вже одержано):

$$\Psi(X,Y,\ldots, Z) \rightarrow \{(x_1,y_1,\ldots,z_1), (x_2,y_2,\ldots,z_2), \ldots, (x_n,y_n,\ldots,z_n)\} \cup \{\Omega\}$$
$$\{(x_1,y_1,\ldots,z_1), (x_2,y_2,\ldots,z_2), \ldots, (x_n,y_n,\ldots,z_n)\} \cap \{\Omega\} = \varnothing$$

Звичайно, система рівнянь є частковим випадком предиката, і в загальнішому випадку рівняння (предикати), які в ньому фігурують, можуть мати різну кількість змінних. Тоді для врахування типів змінних, що входять в предикат, розв'язок можна отримати за правилом перетину відношень, оскільки кожне відношення являє собою таблицю на заданій підмножині змінних:

$$R_1(X,Y,\ldots, Z) \cap R_2(P,Q,\ldots, Z) = R_3(X,Y,P,Q,\ldots, Z)$$

Як правило, реальні системи отримують завдання на зразок: "зроби щось, інакше буде не дуже добре". Ця інформація має дві частини: 1) що потрібно зробити; 2) що саме погане станеться, якщо цього не зробити. Якщо розглядати систему без самоорганізації, то модальності "зроби щось" і "інакше буде не дуже добре" є обов'язковими, тобто робити треба без варіантів. Система, яка має елементи самоорганізації, розглядає всі твердження з модальністю "можливо", а також з істинами відносно деяких суб'єктів. Тобто хтось пропонує комусь "зробити щось", інакше комусь, можливо, "буде не дуже добре". У розвинених інтелектуальних системах частина "буде не дуже добре" не використовується. Система, яка має інтелект, сама дійде до того, що саме буде не добре і кому потрібно щось робити. Аналізуючи модальність "можливо", система повинна зрозуміти, чому саме суб'єкт, який пропонує певне рішення, вважає його найкращим серед усіх інших можливих рішень.

Висновок

Самоорганізація – це основна концепція розвитку системи. Будь-яка система (технічна чи біологічна), яка не самоорганізовується, приречена на відмирання. В статті зроблено спробу представити самоорганізацію програмних систем формально, у вигляді додаткових можливостей перебору варіантів. Поки що це робить програміст, адаптуючи свій програмний продукт під змінні вимоги експлуатації. В результаті порівняння різних алгоритмічних систем – штучних нейронних мереж та декларативних програм ми дійшли висновку, що для виникнення ефекту самоорганізації ці алгоритмічні системи необхідно зблизити. По-друге, фіксована топологія штучної нейронної мережі не дає змоги робити глибокі логічні висновки, як це робить декларативна програма, а декларативна програма має навчитися цілеспрямовано змінювати свій програмний код і вибирати продукцію для виводу за ваговими коефіцієнтами. По-третє, обидві алгоритмічні системи повинні навчитись працювати з наближеною, неповною, неоднорідною та суперечливою інформацією.

1. Neumann von J. Theory of self-reproducing automata (edited by A.W.Burks) – Univ. of Illinois press, Urbana, 1966. – 400 p. 2. Рашкевич Ю., Пелешко Д., Купчак М., Ковальчук А. Виділення квазістаціонарних ділянок мовного сигналу за спектром матричного оператора // Вісник Нац. ун-ту "Львівська політехніка". – 2011. – № 710. – С. 70–74. 3. Frank Rosenblatt, Audio Signal Pattern Perception Device. – US Patent 3287649, Nov. 22, 1966. 4. Candida Ferreira. Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence (Studies in Computational Intelligence). – Springer, 2006. – 498 p.