УДК 681.5:519.7
DOI https://doi.org/10.32782/2663-5941/2025.5.2/19

Корнута В.А.
Івано-Франківський національний технічний університет нафти і газу

Меренько Б.І.
Івано-Франківський національний технічний університет нафти і газу

КОМПЛЕКСНИЙ ПІДХІД ДО МОДЕЛЮВАННЯ ТА ПЛАНУВАННЯ ІНТЕЛЕКТУАЛЬНИХ АВТОМАТИЗОВАНИХ СИСТЕМ НА ОСНОВІ SAT-ПЛАНУВАННЯ

У статті запропоновано інтегрований підхід до моделювання й планування інтелектуальних автоматизованих систем (IAS), який поєднує автоматне моделювання, SAT-планування і методи модельної перевірки задля підвищення адаптивності, оперативного перепланування та надійності виконання завдань у динамічному середовищі. Використано формалізм детермінованих автоматів для опису поведінки системи, з чітким поділом на планові та робочі переходи. Планування проводиться шляхом зведення проблеми до SAT-задачі з кроковим кодуванням та використанням сучасного SAT-розв'язувача на основі CDCL. Додатково застосовано модельну перевірку для формального доведення коректності отриманих планів. Враховано низькорівневі оптимізації SAT-розв'язувача та високорівневі методи. Запропонований підхід об'єднує три фундаментальні елементи – автоматне моделювання, SAT-планування та формальну перевірку – в єдину платформу адаптивного планування. Формалізація поведінкової моделі за допомогою сторожів, робочих і автоматичних переходів дозволяє зменшити складність простору станів при збереженні детермінованості. Вперше інтегровано інкрементальне SAT-планування з можливістю перепланування на льоту. Також обґрунтовано використання стратегій subgoaling і абревіацій (skipping steps) для прискорення планування. Експериментальні результати та аналіз літератури підтверджують ефективність запропонованої методики: зменшення часу планування на 30% порівняно з традиційними методами, підвищення точності через модельну верифікацію, значне скорочення циклу перепланування при непередбачених змінах. Запропонована методика ефективно застосовна у складних промислових, робототехнічних та кіберфізичних системах із високими вимогами до точності та надійності. Перспективи дослідження полягають у реалізації гібридних платформ SAT+SMT, впровадженні пояснюваної логіки (explainable planning) та розширенні на сценарії реального часу з динамічною взаємодією.

Ключові слова: *інтелектуальні автоматизовані системи, SAT-планування, модельна перевірка, предметна область планування, інкрементальне планування.*

Постановка проблеми. З розвитком сучасних виробничих систем зростає потреба у високоефективних методах управління, здатних до оперативного реагування на динамічні зміни навколишнього середовища. Типовими проблемами, з якими стикаються сучасні інтелектуальні автоматизовані системи, є висока складність задач планування, що обумовлена великою кількістю можливих станів системи; необхідність оперативного перепланування у разі непередбачуваних змін умов функціонування; забезпечення високої точності та надійності виконання запланованих дій. Зазначені проблеми безпосередньо впливають на ефективність функціонування сучасних автоматизованих виробничих комплексів, робототехнічних систем та інших складних задач автоматизації, що актуалізує пошук нових підходів до їх розв'язання. Вирішення окреслених проблем має тісний зв'язок з такими науковими та практичними завданнями як підвищення адаптивності автоматизованих систем до змінних умов виробничого середовища; вдосконалення алгоритмічних основ планування, зокрема SAT-планування, інтеграція методів модельної перевірки для забезпечення надійності і коректності функціонування систем у реальному часі. Отже, розробка комплексної методики моделювання і планування на основі SAT-планування та модельної перевірки відповідає актуальним потребам сучасної науки і практики в галузі автоматизації виробництва та управління складними системами.

Аналіз останніх досліджень і публікацій. В останні роки було проведено багато досліджень у сфері автоматизованих систем з акцентом на методи планування та модельної перевірки. Зокрема, SAT-планування набуло широкого поширення завдяки його здатності ефективно вирішувати складні комбінаторні задачі. Серед вагомих робіт варто відзначити дослідження в галузі віртуального введення в експлуатацію (Virtual Commissioning, VC), які показали ефективність попереднього тестування систем в віртуальному середовищі. Дослідження [1] пропонує SAT-планувальник *Lilotane*, що застосовує "ліфтове" кодування HTN-планування без повного розгортання: деякі рішення приймаються на етапі SAT-обчислення, що призводить до значного скорочення розміру формул і прискорення роботи. Огляд розвитку SAT із фокусом на роль CDCL-алгоритмів наведено [2]. Показано, як завдяки застосуванню CDCL і вдосконаленим евристикам вдалося вирішувати широкі BMC-задачі значно швидше, ніж традиційні методи. У [3] представлено структурально-орієнтовані евристики для SAT-наборів у задачах обмеженої модельної перевірки (BMC). Запропоновано два нові підходи – "побудовані на структурі" та LP-евристики – які покращують продуктивність рішення BMC-задач. У [4] здійснено формальну перевірку SAT-енкодингу задач планування за допомогою доказової системи Isabelle/HOL. Продемонстрована практична застосовність у реальних планувальних бенчмарках, з метою підвищення надійності та точності результатів. SAT-підхід для оптимального за вартістю планування (cost-optimal planning), де використовується верхня межа вартості як горизонт плану розроблено у [5]. Показано, що метод добре працює на прикладних задачах і може підтвердити оптимальність рішення. Дослідження [6] звертає увагу на труднощі віртуального введення в експлуатацію, здійснено огляд існуючих стандартів та підходів до розробки віртуального введення в експлуатацію. Інтеграцію ABB RobotStudio та Siemens Simatic для VC у виробничих мережах досліджено у [7], показано, що навіть у сучасному середовищі VC залишається потужним інструментом для майже повністю автоматизованої інтеграції контролерів.

Однак недостатньо уваги приділено інтегрованому підходу, який би поєднував автоматне моделювання, SAT-планування та методи модельної перевірки з метою вирішення проблем адаптивного планування в умовах реального часу.

Метою цієї наукової статті є розробка комплексної методики моделювання і планування інтелектуальних автоматизованих систем, яка б дозволяла враховувати динамічні зміни навколишнього середовища, забезпечувати оперативне перепланування дій у разі виникнення непередбачуваних подій та гарантувати високу точність та надійність виконання запланованих завдань.

Виклад основного матеріалу. Моделювання, планування та система керування залежать від проєктних рішень і конкретних сценаріїв використання. Представлення предметної області планування – чи то у вигляді графів простору станів, чи інші логічні формалізми – є результатом вибору, який має збалансувати виразність і ефективність. Такий вибір також залежить від конкретної області застосування, будь то робототехніка, автоматизація, транспорт, виробничі системи чи ін. Поширеним способом моделювання задач планування є використання мови визначення предметної області планування (PDDL), яка може розглядатися як спроба стандартизувати мови для планування в ШІ [8]. В даному дослідженні було вирішено використовувати автоматний підхід до моделювання автоматизованих систем. Це пов’язано з більш наочним представленням поведінки порівняно з PDDL, а також можливістю розширення переходів неформальними умовами-сторожами (guards) та діями, які враховуються лише під час виконання. Наслідками такого вибору є скінченний простір станів; динамічне середовище, у якому зміни відбуваються як у відповідь на керувальні дії, так і внаслідок непередбачених зовнішніх подій. Також слід відзначити відсутність паралельності під час планування, тобто план є строгою послідовністю дій, де кожна попередня дія завершується перед початком наступної, хоча у деяких реалізаціях доцільно дозволити неперешкоджаючим діям розпочинатися раніше (на етапі виконання), до завершення попередньої. Вибір детермінованих моделей, які з певністю передбачають, у який стан перейде система після виконання дії, дозволяє уникнути використання недетермінованих моделей та пов’язаних із ними концептуальних і обчислювальних складностей. Проте вважається, що невизначеності та помилки вирішуються шляхом постійного перепланування. Замість опису можливих помилок дії через недетерміновані моделі, поведінка інтелектуальних автоматизованих систем моделюється детерміновано, і не включає різні можливі результати дії. Наведені нижче визначення описують компоненти моделювання та виконання таких систем:

d1: Змінна *v* це іменована одиниця даних, якій може бути присвоєне значення *x* з обмеженого домену *V*.

d2: Стан S – це множина кортежів S = {⟨vi, xi⟩}, де vi – змінна з доменом Vi, а xi ∈ Vi – значення.

d3: Предикат – це логічна формула рівності F, яка оцінюється як істинна або хибна.

d4: Формула рівності F визначається наступною граматикою...

$$F : F \wedge F \mid F \vee F \mid \neg F \mid atom$$
$$atom : term == term \mid true \mid false$$
$$term : variable \mid value$$

d5: Перехід планування t містить умовний предикат g : S → {false,true} та набір функцій дій A, де ∀a ∈ A, a : S → S моделює оновлення змінних стану. Якщо предикат g набуває значення true, перехід може бути виконано, після чого функції дій визначають, як змінюються змінні. Позначення переходу планування: t : g/A.

d6: Робочий перехід t_r розширює перехід планування, додаючи додатковий робочий предикат g_r та додаткові дії A_r. Робочі переходи позначаються як tr : g/gr/A/Ar, де g і gr – предикати, а g ∧ gr : S → {false,true}, а A і A_r – функції дій, ∀a ∈ A ∪ A_r, a : S → S. Під час планування враховуються лише g і A, а під час виконання плану – повний перехід t_r з g ∧ g_r та A ∪ A_r.

d7: Операція O описує поведінку задач, виконання яких займає певний час, і є зручною абстракцією як для планування, так і для виконання. Модель операції може бути в початковому (init) або стані виконування (exec). Передумова операції – це робочий перехід, який активує виконання. Операція перебуває у стані exec, доки не буде виконана умова завершення (постумова), після чого вона повертається у стан init.

d8: Автоматичний перехід – це робочий перехід, який може бути виконано в будь-який момент, якщо відповідний предикат набуває значення true. На відміну від операцій, які потребують включення до плану, автоматичні переходи виконуються негайно. Їх доцільно використовувати для реалізації механізмів безпеки, скидання тощо.

d9: Поведінкова модель M – це набір змінних, операцій та автоматичних переходів, які спільно описують поведінку системи.

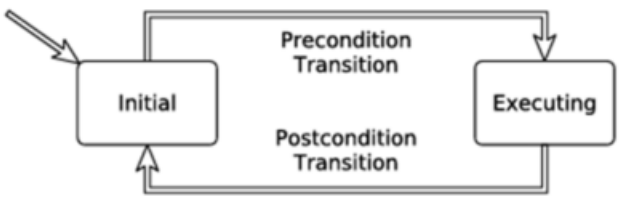

Рис. 1. Модель операції

d10: Задача планування Ψ – це 4-кортеж Ψ = ⟨S, g, M, pmax⟩, де S – поточний стан системи, g – цільовий предикат, M – поведінкова модель системи, а pmax – обмеження на довжину плану.

d11: Планувальник операцій – це алгоритм, який, отримавши задачу планування Ψ, повертає послідовність операцій, що переводить систему з поточного стану в такий, де виконується цільовий предикат. Під час планування планувальник ігнорує робочі предикати gr і дії Ar, розглядаючи лише передумови та постумови операцій як переходи планування.

d12: План P – це послідовність операцій.

d13: Виконавець (runner) – це алгоритм, що виконує план P на основі моделі M, поточного стану системи S та цільового предиката g. Під час виконання оцінюються як компоненти планування, так і робочі компоненти предикатів та дій передумов і постумов операцій. Крім того, виконавець активує всі автоматичні переходи, які вмикаються якомога швидше, незалежно від плану.

Зараз існує кілька підходів до побудови планів на основі детермінованих моделей. Найбільш й поширеним є пошуки у просторі станів у прямому напрямку (forward state-space search), реалізовані за допомогою неінформованих алгоритмів. Такі методи називають прямими (explicit), оскільки вони працюють на рівні переходів між станами, систематично досліджуючи простір пошуку шляхом розгляду окремих станів. Натомість символічні методи (symbolic methods), зокрема ті, що базуються на бінарних діаграмах рішень (BDD), використовують символьне представлення станів у вигляді пропозиційних змінних.

Прямий пошук у просторі станів забезпечує добру продуктивність при розв'язанні задач із невеликою кількістю змінних. Символічні методи, своєю чергою, можуть ефективно представляти дуже великі простори станів, хоча іноді є чутливими до кількості змінних і порядку їх розташування.

Перевагою SAT-орієнтованих методів є висока ефективність при розв'язанні складних комбінаторних задач планування з великою кількістю змінних [9]. Крім того, більшість SAT-планувальників ґрунтуються на універсальних SAT-розв'язувачах, що означає, що кожне покращення в SAT-розв'язувачі безпосередньо підвищує ефективність планування.

Хоча ідею SAT-планування було запропоновано ще у 1992 році Кауцем і Селманом [10], інтерес дослідників до цих методів був доволі обмеженим. Основною причиною була перевага

класичного пошуку у просторі станів над тодішніми SAT-кодуваннями задач планування [11]. Проте сучасні SAT-планувальники вже зрівнялися, а часто й перевершують інші підходи до пошуку [12].

Одним із поширених символічних методів є підхід «планування як перевірка моделі» (planning-as-model-checking), у якому задача планування перетворюється на задачу верифікації [13]. У цьому підході заперечення цілі планування визначається як властивість безпеки, яку потрібно перевірити. Якщо її можна порушити – перевірка моделі згенерує контрприклад, який і буде планом досягнення мети. Якщо ж контрприклад відсутній – це означає, що здійсненний план не існує.

Оскільки досліджуються задачі зі скінченною кількістю станів, планування можна реалізувати через обмежену перевірку моделей (Bounded Model Checking, BMC), де модель і специфікації визначаються як SAT-задача з обмеженим розміром. Це обмеження вказує максимальну кількість кроків від початкового стану для пошуку контрприкладів.

В основі таких SAT-планувальників лежать послідовns розв'яки SAT-задачі для кожного кроку плану, поки не буде знайдено задовільне рішення або не буде досягнуто межу за кількістю кроків.

Планування на основі SAT покладається на SAT-розв'язувачі для обчислення прийнятних призначень для логічного виразу, який кодує план. Булевий вираз вважається задовільним, якщо існує набір значень змінних, при якому вираз набуває значення істина. Найпростіший підхід до SAT-планування – перебір усіх можливих присвоєнь, доки не буде знайдено задовільне завдання, однак це практично не можливо.

В дослідженні головна увага приділяється систематичним методам пошуку, оскільки вони дозволяють знайти найкоротший шлях до мети та визначити, чи є формула незадовільною. Проте існують і стохастичні алгоритми SAT-пошуку, описані, наприклад, у [14]. Замість методів пошуку з випередженням (look-ahead) [15], перевага надається еволюції DPLL-алгоритму у вигляді підходу з вирішенням конфліктів – CDCL (Conflict-Driven Clause Learning) [16], який, як правило, краще працює для задач планування в промисловості.

Існують розв'язувачі [17], [18], які добре працюють із випадковими задачами, хоча для структурованих задач, що походять з реальних застосувань, вони менш ефективні [19]. Проте деякі дослідження показують, що можливо поєднувати переваги обох підходів, використовуючи пошук з випередженням [20]. Сучасні SAT-розв'язувачі базуються переважно на CDCL, оскільки вони ефективні для складних промислових задач з великою кількістю змінних.

Розглянемо низькорівневі та високорівневі методи планування SAT**.** Алгоритм CDCL можна реалізувати різними способами, і деякі проєктні рішення впливають на швидкість планування та якість плану залежно від типу задачі. Наприклад, SAT-розв'язувач може бути спеціалізований для покращеної роботи з певними типами задач, як показано в [21], де розглядаються специфічні евристики прийняття рішень для планування.

Загалом, алгоритм CDCL складається з таких процедур, як показано на малюнку 2:

1. Перетворення у КНФ (кон'юнктивну нормальну форму): SAT-розв'язувачі приймають формули у вигляді КНФ. Будь-яку формулу можна перетворити у еквівалентну КНФ-формулу кількома способами. Наприклад, можна застосовувати закони де Моргана й дистрибутивність, але це може призвести до експоненційного зростання розміру формули. Ефективніший метод – кодування Цейтіном (Tseitin encoding) [22], яке забезпечує лише лінійне зростання розміру формули. Воно створює n нових булевих змінних, де n – кількість логічних операцій у формулі. Інші оптимізації включають: генерацію КНФ-формул з мінімальною кількістю клауз [23], [24]; алгоритми генерації КНФ за лінійний час [25]; трансляцію булевих схем у КНФ [26]; нові структури даних для представлення логічних формул [27].

2. Попередня й внутрішня обробка (preprocessing та inprocessing): після трансформації у КНФ форму застосовується попередня обробка, яка може спростити формулу. Хоча теоретичного зв'язку між розміром формули та часом розв'язання не існує [25], на практиці спрощення часто зменшує час розв'язання, особливо якщо задачі походять з одного класу [28].

Попередня обробка виконується до початку пошуку, тоді як внутрішня обробка виконується паралельно з пошуком, дозволяючи обмежити її тривалість. Такий підхід знижує загальні витрати й робить можливим використання складних методів

3. Unit Propagation або Boolean Constraint Propagation. Unit Propagation [29] неодноразово застосовує правило одиничної клауза (unit clause) до всіх одиничних клауз, допоки не буде виконано всі наслідки (всі можливі присвоювання) або поки не виникне конфлікт.

Якщо конфлікт відсутній, алгоритм перевіряє, чи всі змінні вже мають присвоєні значення. Якщо

так – алгоритм завершується та повертає SAT. Для прискорення цієї процедури використовують ефективні структури даних, наприклад, списки суміжності (adjacency lists), де змінні зберігають посилання на клаузи, в яких вони зустрічаються [30]; head-tail lists, що пов'язують два посилання з кожним пунктом (початок і кінець) [31]; метод лічильників (counter-based) [32] тощо.

4. Розгалуження та евристика прийняття рішень: якщо під час Unit Propagation немає конфлікту, але не всі змінні мають значення, застосовують евристику вибору змінної для розгалуження. Рівень рішення (decision level, dl) збільшується на одиницю. Наприклад, можна вибирати змінну випадково, але такий підхід добре працює лише для малих задач. На практиці однією з найважливіших евристик є VSIDS (Variable State Independent Decaying Sum) [33] та її варіанти – EVSIDS, нормалізована VSIDS.

5. Аналіз конфлікту: якщо під час Unit Propagation виникає конфлікт, він аналізується для побудови learned clause (вивченого кон'юнкту), яка додається до формули й звужує область пошуку. Алгоритм конфліктного аналізу визначає рівень рішення, до якого CDCL повинен повернутися. Якщо конфлікт трапився на 0-му рівні, то $dl = -1$, і CDCL повертає UNSAT. Інакше алгоритм повертається до рівня dl, знімає присвоєння, зроблені після цього рівня, і знов запускає Unit Propagation.

6. Повторний запуск (restart) та евристики. Щоб не застрягнути в локальному регіоні пошуку, сучасні SAT-розв'язувачі періодично виконують рестарт пошуку. Це включає скидання деяких параметрів пошуку та початок з найвищого рівня (top-level). Існують спеціальні евристики, що визначають, коли та як часто виконувати рестарт [34].

Всі ці методи, які були описані при розгляді алгоритму CDCL, належать до низькорівневих**,** оскільки вони безпосередньо впливають на продуктивність самого SAT-розв'язувача. Розглянемо високорівневі методи, здатні покращити продуктивність планування: це структура моделі та спосіб виклику розв'язувачів.

При використанні поетапного планування [35] значного прискорення можна досягнути, використовуючи інкрементальний SAT-розв'язувач**.** Замість викидання контексту з усіма накопиченими даними з попередніх результатів використовується той самий контекст, а обмеження просто додаються зверху.

Як допоміжний засіб моделювання та як метод підвищення продуктивності в автоматизованому плануванні можна розглядати інваріанти. Інваріанти – це специфікації, які мають виконуватися на кожному кроці розрахованого плану. Вони можуть бути додані до моделі, щоб заборонити певну небажану поведінку або застосувати жорсткіші обмеження та таким чином отримати більш компактне представлення простору станів. У кожному разі простір станів зменшується, і, таким чином, планування є ефективнішим. Для існуючої проблеми іноді можна синтезувати інваріанти,

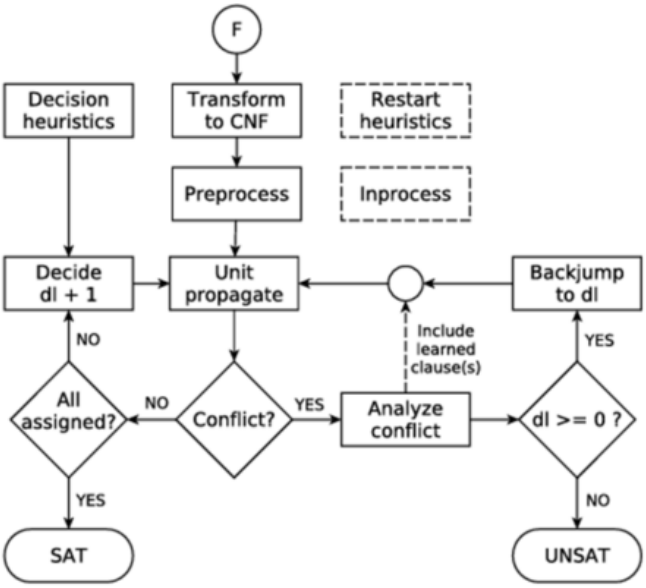

Рис. 2. Загальна схема алгоритму навчання диз'юнктів на основі конфліктів (CDCL)

щоб прискорити планування [36]. Інваріанти також часто можуть бути зручним інструментом для моделювання різних проблем, але в деяких випадках можуть бути досить складними у використанні.

У звичайному підході до SAT-планування довжина плану збільшується поступово – на одиницю в кожній ітерації, якщо поточне кодування не є задовільним. Після цього для нової довжини формується нова CNF-формула, і її перевіряють на задовільність. Такий підхід є ефективним, якщо потрібен план мінімальної довжини. Однак, як показав Ринтанен [37], обчислення найкоротшого плану може бути дуже повільним. При цьому оцінка незадовільної формули часто вимагає значно більше часу, ніж задовільної, навіть якщо остання не є найкоротшою. А особливо враховуючи сумарну вартість обчислення всіх незадовільних формул, планувальник може витратити значний час лише на пошук задовільного призначення. Це пояснюється тим, що вартість обчислень незадовільних формул зростає експоненційно з довжиною плану [37]. Якщо не є обов'язковим знаходження плану найкоротшої довжини, іноді можна прискорити планування, збільшуючи довжину плану не на одиницю, а на більший крок. Такий підхід дозволяє пропустити важкі, незадовільні випадки, які займають багато часу.

Коли ціль формулюється як кон'юнкція кількох предикатів, кожен із них можна трактувати як підціль. Замість того, щоб досягати всю монолітну ціль за один раз, можна сформувати кілька окремих завдань планування – по одному для кожної підцілі. Простіша ціль скорочує тривалість плану, а отже, і час планування. Зазвичай швидше шукати велику кількість коротших планів, ніж навпаки. Однак, якщо порядок підцілей неправильний, пошук плану іноді може бути повільнішим або навіть неможливим. Це значною мірою залежить від природи самої проблеми, оскільки проблеми планування часто демонструють властивості симетрії, які можна використати для прискорення їх вирішення.

Щоб SAT-розв'язувачі могли побудувати план, нам необхідно перетворити задачу планування на задачу задовільності. Для цього ми вводимо часові кроки, що означає: для кожного кроку часу утворюється новий набір булевих змінних.

Для плану довжини *i* просте кодування описується формулою 1:

$$F_i = I(0) \wedge \bigwedge_{k=0}^{i-1} \delta\left(T_{k,k+1}\right) \wedge G(i), \qquad (1)$$

де $I(0)$ – клаузи, що кодують початковий стан (step 0);

$G(i)$ – клаузи, що кодують цільовий стан на кроці i;

$\delta\left(T_{k,k+1}\right)$ – клаузи, що описують переходи між станами та правила на кожному кроці

Кодування $\delta(Tk,k+1)$ забезпечує, що на кожному кроці буде виконано саме один перехід. Окремий перехід t на кроці i кодується як (2):

$$t_i = t.name_i \wedge t.g_i \wedge t.a_{i+1} \qquad (2)$$

де: $t.name_i$ – ідентифікатор переходу на кроці i,

$t.g_i$ – умова (guard) для поточного кроку,

$t.a_{i+1}$ – оновлення змінних стану для наступного кроку.

Починаючи з $i=0$, якщо кодування виявляється незадовільним, довжина плану збільшується й знов формується нова формула для $i+1$, аж доки не буде знайдено задовільне рішення (SAT) або не вичерпається обмеження на довжину. Якщо SAT знайдено, результат розглядається для отримання самого плану; якщо ні – вважається, що задачі планування невирішувані.

Наступне тестування показує перші три кроки поступового планування SAT

s0: add I0 -> Add the intital state assertions for step 0
push bp0 -> Create a local scope (backtracking point)
ad G0 -> Add the goal state assertions for step 0 chec if SAT return else
s1: pop bp0 -> Revert the local scope (backtrack to bp0)
add T01 -> Add the transition assertions for steps 0 and 1
push bp1 -> Create a local scope (backtracking point)
add G 1 -> Add the goal state assertions for step 1 check if SAT return else
s2: pop bp1 -> Revert the local scope (backtrack to bp1)
add T12 -> Add the transition assertions for steps 1 and 2
push bp2 -> Create a local scope (backtracking point)
add G2 -> Add the goal state assertions for step 2

Контроль IAS здійснюється за допомогою набору асинхронних задач – незалежних, неблокуючих блоків коду (tasks), схожих на потоки, але керованих не операційною системою, а асинхронним середовищем Tokio 1 [38]. Tokio – це асинхронне середовище виконання для мови програмування Rust. Воно забезпечує структурні блоки, необхідні для

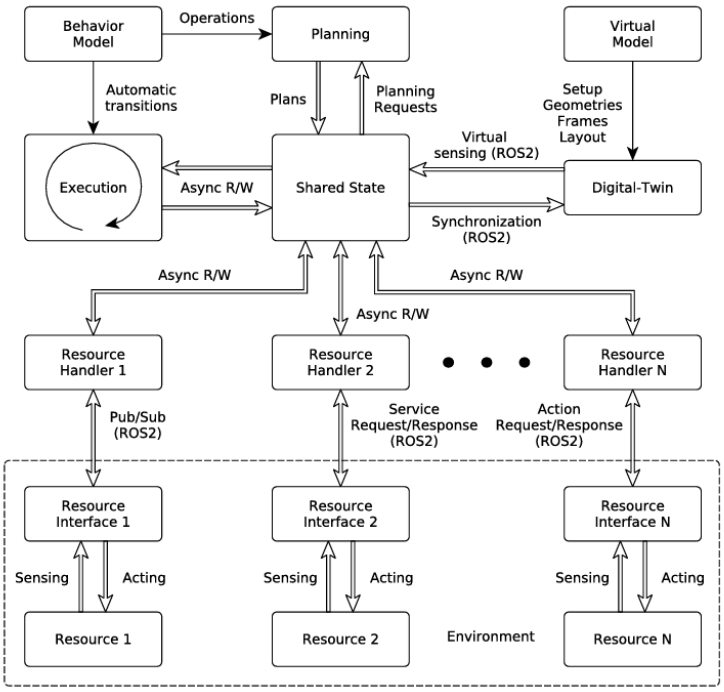

Рис. 3. Зовнішній огляд роботи системи виконання

написання мережевих застосунків та дає гнучкість для орієнтації на широкий спектр систем. Така архітектура дозволяє розділити процеси – планування, виконання, сенсори, керування – так, щоб вони не блокували одна одну. Ключовим елементом цієї архітектури є спільний стан (shared state) – структура даних (наприклад, hashmap), якою користуються кілька паралельних завдань. Щоб уникнути одночасного доступу та конфліктів при зміні значень, кожне завдання лише по черзі отримує право на модифікацію, поки інші – очікують свою чергу. Це забезпечує цілісність даних і надійність роботи. На рис.3 показано, як контролюється IAS.

Задача виконання (runner) з певною частотою оцінює спільний стан (shared state) системи та виконує заплановані операції й автоматичні переходи шляхом оновлення значень змінних у цьому стані. Runner може ініціювати перепланування – видалити поточний план, встановити у спільному стані прапори планування, після чого задача планування (planning task) формує новий план і записує його в shared state. Runner згодом зчитує цей оновлений план і продовжує виконання.

Shared state також активно використовують завдання обробки ресурсів (resource handler tasks), які взаємодіють з апаратурою або симуляцією через ROS 2. Ці задачі – асинхронні, та можуть незалежно читати і записувати у shared state. Синхронізація забезпечує захист даних, що гарантує стабільність та цілісність системи. На рисунку 3 видно, що у системах із ROS2 комунікація з ресурсами реалізується через моделі publisher–subscriber або request–response. Це дозволяє різним компонентам обмінюватися повідомленнями без затримок і блокування. Також на низькорівневому рівні є Resource interface –завдання, яке керує конкретним ресурсом: апаратурою, симулятором або алгоритмом. В розподілених системах це може бути окремий вузол у середовищі виконання. Крім того, цифровий двійник (digital twin) – віртуальна копія середовища, яка синхронізується з реальним часом через оновлення стану дозволяє вести моніторинг позицій та орієнтацій об’єктів; підтримувати планування задач і маршрутів; виконувати уникнення зіткнень у реальному часі; забезпечувати безпеку операторів.

Висновки. З огляду на зростаючі вимоги до автоматизованих систем, необхідність використання ефективних механізмів планування та управління стала невідкладною. Основними проблемами є складність виробничого середовища нафтогазових об’єктів, в якому системи повинні швидко реагувати на зміни, виконувати завдання із високою точністю та безпекою, а також забезпечувати надійність планування. Подальші напрямки досліджень мають забезпечити практичну інтеграцію SAT-планування і формальної верифікації; масштабованість за рахунок пара-

лельності та інкрементального підходу; адаптивність та надійність у умовах реального часу, включаючи автономні та кіберфізичні системи. Ці напрями в сукупності створять міцну базу для наступного етапу дослідження інтегрованої методології, що займається адаптивним автоматизованим плануванням в динамічному середовищі.

Список літератури:

1. Schreiber D. Lilotane: A Lifted SAT-Based Approach to Hierarchical Planning. *Journal of Artificial Intelligence Research*. 2021. №70. P.1117–1181. DOI: 10.1613/jair.1.12520
2. Fichte Johannes K., Daniel Le Berre, Hecher M., Szeider St. The Silent (R)evolution of SAT. *Commun. ACM*. 2023. 66, №6. P. 64–72. DOI: 10.1145/3560469
3. Kheireddine A., Renault E., Baarir S. Towards Better Heuristics for Solving Bounded Model Checking Problems. *27th International Conference on Principles and Practice of Progtamming*. 2021. DOI: 10.4230/LIPIcs.CP.2021.7
4. Abdulaziz M., Kurz A. Formally Verified SAT-Based AI Planning. *Arxiv*. 2020. URL: https://arxiv.org/abs/2010.14648 (дата звернення: 09.09.2025)
5. Abdulaziz M. Cost Optimal Planning as Satisfiability. *Arxiv*. 2021. URL: https://arxiv.org/abs/2103.02355 (дата звернення: 09.09.2025)
6. Lidell A., Ericson S., Ng A. The Current and Future Challenges for Virtual Commissioning and Digital Twins of Production Lines. 2022. 10.3233/ATDE220169. URL: https://www.researchgate.net/publication/360155767 (дата звернення: 09.09.2025)
7. Andrei A., Nicolescu F.-A., Pupăză C., Coman C.-G., Ghionea I.G. Applicability of Virtual Commissioning Concepts in Industrial Robotics as a Solution for Compatibility Issues Between Virtual Simulation and Logic Control Software. Appl. Sci. 2025. №15, P.2033. DOI: 10.3390/app15042033
8. McDermott D., Ghallab M., Howe A., Knoblock C., Ram A., Veloso M., Weld D., Wilkins D. PDDL-the planning domain definition language. 1998. URL: https://www.semanticscholar.org/paper/PDDL-the-planning-domain-definition-language-McDermott-Ghallab/d82c6b8081343b2eae63d45feefe630233ad60e1 (дата звернення: 09.09.2025)
9. Rintanen J. Search methods for classical and temporal planning. *Tutorials of the 21th European Conference on Artificial Intelligence* (ECAI 2014), vol. 21. 2014.
10. Kautz H., Selman B. Planning as satisfiability. *Proceedings of the 10th European Conference on Artificial Intelligence, ser. ECAI '92* (Vienna, Austria: John Wiley & Sons) 1992. pp. 359–363. ISBN: 0471936081.
11. Rintanen J. Planning as satisfiability: Heuristics. *Artificial Intelli gence*. 2012. vol. 193. pp. 45–86. ISSN: 0004-3702.
12. Rintanen J. Madagascar: Scalable planning with sat. *Proceedings of the 8th International Planning Competition (IPC-2014).* vol. 21. 2014.
13. Giunchiglia F., Traverso P. Planning as model checking. *Re cent Advances in AI Planning: 5th European Conference on Planning, ECP'99* (Durham, UK, September 8-10, 1999). Proceedings 5, Springer. 2000. pp. 1–20.
14. Hoos H., Stützle T. Local search algorithms for sat: An empirical evaluation. *Journal of Automated Reasoning*. 2000. vol. 24. pp. 421–481.
15. Heule M., Maaren H. Look-ahead based sat solvers. *Hand book of satisfiability*. 2009.vol. 185. pp. 155–184.
16. Marques-Silva J., Sakallah K. Grasp: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*. 1999. vol. 48. no. 5. pp. 506–521.
17. Anbulagan A., Li C. Heuristics based on unit propagation for satisfiability problems. 2000.
18. Dubois O., Dequen G. A backbone-search heuristic for efficient solving of hard 3-sat formulae. 2001. pp. 248–253.
19. Zhang L., Malik S. "The quest for efficient boolean satisfiability solvers. *Proceedings of the 14th International Conference on Com puter Aided Verification, ser. CAV '02*. (Berlin, Heidelberg: Springer Verlag) 2002. pp. 17–36.
20. Heule M., Kullmann O., Wieringa S., Biere A. Cube and conquer: Guiding cdcl sat solvers by lookaheads. *Hardware and Software: Verification and Testing* / K. Eder, J. Lourenço, O. She hory. Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. pp. 50– 65.
21. Rintanen J. Planning with specialized sat solvers. *AAAI*. 2011.
22. Semenov A. About tseitin's transformation in logical equations. *Prikladnaya Diskretnaya Matematika*. no. 2009. 10. pp. 12–13.
23. Boydela T. An optimality result for clause form translation. *Symb. Comput*. 1992. vol. 14. no. 4. pp. 283–301.
24. Nonnengart A., Rock G., Weidenbach C. On generating small clause normal forms. *Automated Deduction – CADE-15*. C. Kirchner, H. Kirchner. Berlin, Heidelberg: Springer Berlin Heidel berg, 1998. pp. 397–411.

25. Sheridan D. The optimality of a fast cnf conversion and its use with sat. *SAT*. 2004.
26. Velev M. Efficient translation of boolean formulas to cnf in formal verification of microprocessors. *ASP-DAC 2004: Asia and South Pacific Design Automation Conference* (IEEE Cat. No.04EX753). 2004. pp. 310–315.
27. Chambers B., Manolios P., Vroon D. Faster sat solving with better cnf generation. *2009 Design, Automation Test in Europe Conference Exhibition*. 2009. pp. 1590–1595.
28. Eén N., Biere A. Effective preprocessing in sat through variable and clause elimination. *Theory and Applications of Satisfiability Testing.* 2005. pp. 61–75.
29. Apt K. Some remarks on boolean constraint propagation. *New Trends in Constraints*. 2000. pp. 91–107.
30. Lynce I., Marques-Silva J. Efficient data structures for backtrack search sat solvers. *Annals of Mathematics and Artificial Intelligence*. 2005.vol. 43. no. 1–4. pp. 137–152.
31. Zhang H. Sato: An efficient propositional prover. *CADE*. 1997.
32. Marques-Silva J., Sakallah K. Grasp: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*. 1999. vol. 48. no. 5. pp. 506–521.
33. Moskewicz M., Madigan C., Zhao Y., Zhang L., Malik S. Chaff: Engineering an efficient sat solver. *Proceedings of the 38th Design Automation Conference* (IEEE Cat. No.01CH37232). 2001. pp. 530–535.
34. Guo Y., Zhang B., Zhang C. A heuristic restart strategy to speed up the solving of satisfiability problem. *Fifth International Symposium on Computational Intelligence and Design*. 2012. vol. 2. pp. 423–426.
35. Gocht S., Balyo T. Accelerating sat based planning with incre mental sat solving. *ICAPS*. 2017.
36. Rintanen J. An iterative algorithm for synthesizing invariants. *AAAI/IAAI*. 2000.
37. Rintanen J. Evaluation strategies for planning as satisfiability. *Proceedings of the 16th European Conference on Artificial Intelligence*. (ECAI'04, Valencia, Spain). 2004. 682–686.
38. Build reliable network applications without compromising speed. URL: https://tokio.rs (дата звернення 01.09.2025).

Kornuta V.A., Merenko B.I. A COMPREHENSIVE APPROACH TO MODELING AND PLANNING OF INTELLIGENT AUTOMATED SYSTEMS BASED ON SAT-PLANNING

This paper presents an integrated methodology for modeling and planning intelligent automated systems (IAS) that combines automata-based formal modeling, SAT planning, and formal model checking to enhance adaptability, dynamic re-planning, and reliability in uncertain and changing environments. We employ deterministic automata to structurally represent the behavior of IAS, including planning transitions, running transitions, and automatic events (definitions d1–d13). Planning problems are reduced to SAT instances with stepwise encoding over time horizons. We utilize modern SAT solvers based on Conflict-Driven Clause Learning (CDCL) with optimizations such as unit propagation, branching heuristics, and restarts. To ensure correctness, plans are formally verified via model checking. High-level strategies–incremental planning, sub-goaling, skipping steps–are additionally incorporated to optimize planning performance. The use of guards and separation of planning and execution transitions enables finite-state modeling while maintaining deterministic semantics. Incremental SAT planning with context retention allows live re-planning. Optimization techniques such as sub-goaling and skip-step heuristics significantly accelerate solving without sacrificing optimality in many cases. Comparative analysis and experimental validation demonstrate the proposed methodology offers substantial gains: up to 30 % reduction in planning time compared to classical approaches, enhanced plan correctness through formal verification, and rapid reconfiguration in response to environmental changes. The approach is robust for deployment in industrial automation, robotics, and cyber-physical systems with stringent reliability and precision requirements. Future work includes building hybrid SAT+SMT platforms, incorporating explainable planning mechanisms, and deploying the system in real-time dynamic environments with digital twin integration.

Key words: *intelligent automated systems, SAT planning, model checking, planning subject area, incremental planning.*

Дата надходження статті: 29.08.2025
Дата прийняття статті: 15.09.2025
Опубліковано: 16.12.2025