**УДК 004.93:681.32**

**О. Гожий, О. Маленовський**
Чорноморський державнинй університет ім. Петра Могили,
кафедра інформаційних технологій і програмних систем

# АНАЛІЗ ЕФЕКТИВНОСТІ БАГАТОКРИТЕРІАЛЬНИХ ГЕНЕТИЧНИХ АЛГОРИТМІВ У ЗАДАЧАХ ПРИЙНЯТТЯ РІШЕНЬ



**Розглянуто питання використання багатокритеріальних генетичних алгоритмів в задачах прийняття рішень. Були детально досліджені алгоритми: NSGA-II, AMGA-2 та ε-MOEA. На тестовому прикладі було розглянуто обчислювальну складність алгоритмів та визначені переваги і недоліки їх використання.**

**Ключові слова: багатокритеритеріальні генетичні алгоритми, задачі прийняття рішень.**

**The article deals with the use of genetic multi-objective algorithms in problems of decision making. Were investigated algorithms: NSGA-II, AMGA-2 and ε-MOEA. In the test case was considered computational complexity of algorithms and identified advantages and disadvantages of their use.**

**Key words: genetic multi-objective algorithms, problems of decision making.**

## 1. Постановка проблеми

В останні роки тема багатокритеріальної оптимізації та багатокритеріальних методів прийняття рішень набула значної популярності, було запропоновано багато нових методів та алгоритмів для розв'язання багатокритеріальних задач у різних галузях. Використання багатокритеріальної оптимізації стимулювалося за рахунок появи швидкодійних обчислювальних систем і моделей чисельного аналізу для вирішення різноманітних інженерних проблем. Значного розвитку набули обчислювальні методи і алгоритми, які базуються на еволюційних принципах, вони знайшли широке застосування, адже більшість інженерних задач характеризуються NP-складністю, тому часто бажаним є швидке обчислення наближених розв'язків. Еволюційні алгоритми (ЕА) є техніками адаптивного пошуку, які базуються на природних принципах. Адаптивна природа ЕА використовувалася для розробки алгоритмів оптимізації шляхом створення відповідних операторів варіації і апроксимованих функцій придатності. Генетичний алгоритм (ГА) – це одна з еволюційних технік, яку можна успішно використовувати як інструмент оптимізації. Як правило, ГА працює з популяцією (набором розв'язків), а не з одним розв'язком (індивідом). Підхід на основі популяції у ГА робить його стійким до передчасного сходження, тобто це потужний інструмент для роботи з нелінійними і мультимодальними функціями. Більшість дослідницьких робіт у сфері багатокритеріальних еволюційних підходів представляють і порівнюють різні алгоритми.

У цій статті основна увага приділяється головним питанням розробки багатокритеріальних ГА, основним особливостям використання багатокритеріальних ГА для розв'язання задач багатокритеріальної оптимізації, а також вибір і порівняння найефективніших з них. Зазначимо, що незважаючи на існування вже відомих алгоритмів, багато дослідників, які застосовували багатокритеріальні ГА для виконання певних завдань, проектували свої власні алгоритми шляхом адаптації стратегій з різних багатокритеріальних ГА.

### Аналіз багатокритеріальних генетичних алгоритмів

Загальний ГА з однією ціллю можна модифікувати для пошуку набору декількох розв'язків, над якими не домінують інші, за один прохід. Здатність ГА паралельно шукати у різних місцях області розв'язків робить можливим знаходження набору різних розв'язків складних задач з неопуклими, розривними і мультимодальними областями розв'язків. Оператор кросоверу в ГА дозволяє обробляти структури добрих розв'язків з урахуванням різних цілей для того, щоб

створювати нові недомінуючі розв'язки у недосліджених областях фронту Парето. До того ж більшість багатокритеріальних ГА не вимагають від користувача встановлення пріоритетів, масштабів і ваг для цілей. Тому підхід на основі ГА став найпопулярнішим евристичним підходом до розв'язання задач багатокритеріальної оптимізації та дизайну. 90 % підходів до багатокритеріальної оптимізації спрямовані на апроксимацію істинного фронту Парето для основної задачі. Більшість із них використовують мета-евристичну техніку, і 70 % усіх мета-евристичних підходів базуються на еволюційному підході[1–4].

Перший багатокритеріальний ГА, названий Vector Evaluated GA (VEGA) був запропонований в роботі [5]. Пізніше було розроблено інші багатокритеріальні еволюційні алгоритми, наприклад, такі, як Multi-objective Genetic Algorithm (MOGA) [9], Niched Pareto Genetic Algorithm (NPGA) [13], Weight-based Genetic Algorithm (WBGA) [6], Random Weighted Genetic Algorithm (RWGA)[6], Non-dominated Sorting Genetic Algorithm (NSGA) [13, 15], Strength Pareto Evolutionary Algorithm (SPEA) [6], покращений SPEA (SPEA2) [10], Pareto-Achieved Evolution Strategy (PAES) [6], Pareto Envelope-based Selection Algorithm (PESA) [6], Region-based Selection in Evolutionary Multiobjective Optimization (PESA-II) [9], Fast Non-dominated Sorting Genetic Algorithm (NSGA-II) [11, 13, 14], Multi-objective Evolutionary Algorithm (MEA) [9], Micro-GA, Rank-Density Based Genetic Algorithm (RDGA) [7] і Dynamic Multi-objective Evolutionary Algorithm (DMOEA) [6].

Як правило, багатокритеріальні ГА відрізняються один від одного процедурами розрахунку придатності, підходами до елітарності і диверсифікації. У таблиці подано інформацію про недоліки та переваги загальновідомих ГА[2] .

**Порівняльна характеристика відомих ГА**

| Алгоритм | Призначення придатності | Механізм урізноманітнення | Елітизм | Зовнішня популяція | Переваги | Недоліки |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| VEGA | Розрахунок для підмножин популяції відповідно до різних цілей | Нема | Нема | Нема | Перша проста реалізація багатокритеріального ГА | Тенденція сходження до крайніх значень цільових функцій |
| MOGA | Ранжування за Парето | Створення ніш для розподілу за значенням придатності | Нема | Нема | Просте розширення ГА з однією ціллю | Зазвичай повільне сходження. Проблеми параметру розміру ніші. |
| WBGA | Зважене середнє нормалізованих цільових функцій | Ніши. Наперед визначені ваги. | Нема | Нема | Просте розширення ГА з однією ціллю | Складнощі з не опуклими областями значень функції мети |
| NPGA | Відсутнє призначення придатності, відбір на основі турніру | Підрахунок ніш для розбиття вузлів при виборі за турніром. | Нема | Нема | Дуже простий процес відбору за турніром. | Проблеми параметру розміру ніші. Необхідність додаткового параметру для відбору за турніром. |
| RWGA | Зважене середнє нормалізованих цільових функцій | Привласнення випадкових значень вагам. | Так | Так | Ефективна і проста реалізація. | Складнощі з не опуклими областями значень функції мети |
| PESA | Відсутнє призначення придатності | Коміркова щільність | Так (чисто елітарний) | Так | Проста реалізація. Швидкий алгоритм. | Продуктивність залежить від розміру комірок. Необхідна додаткова попередня інформація про область рішень. |

*Продовження таблиці*

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| PAES | Використовується домінантність Парето для заміщення пращура, якщо нащадок домінує. | Коміркова щільність для розбиття вузлів між нащадками і пращурами. | Так | Так | Спеціальна стратегія для випадкової мутації. Легка реалізація. Ефективність. | Підход не базується на популяціях. Продуктивність залежить від розміру комірок. |
| NSGA | Ранжування на основі не домінантного сортування | Створення ніш для розподілу за значенням придатності | Нема | Нема | Швидке сходження. | Проблеми параметру розміру ніші. |
| NSGA-II | Ранжування на сонові не домінантного сортування | Метод групової відстані | Так | Нема | Єдиний параметр. Перевірений та ефективний алгоритм. | Групова відстань має сенс лише в цільовій області |
| SPEA | Ранжування на основі зовнішнього сховища недомінантних розв'язків | Кластеризація для обмеження зовнішньої популяції | Так | Так | Перевірений алгоритм. Відсутність параметру для кластеризації. | Складний алгоритм кластеризації. |
| SPEA-2 | Міцність домінантів | Щільність, яка обраховується на осноі даних ро к-ого сусіда | Так | Так | Покращений SPEA. Гарантоване збереження точок екстремумів. | Затратні з точки зору обчислення процедури розрахунку придатності і щільності. |
| RDGA | Задача зводиться до задачі з двома цілями – ранжуванням розв'язків і щільністю | Заборона локальної коміркової щільності | Так | Так | Динамічне оновлення комірок. Надійний з огляду на кількість цілей | Складний в реалізації. |

**Обґрунтування вибору методів NSGA-II, AMGA-2 та ε-MOEA**

Для перевірки обчислювальної ефективності застосування багатокритеріальних алгоритмів в задачах прйняття рішень було реалізовано такі алгоритми: **NSGA-II**, **AMGA-2** та **ε-MOEA**. Цей вибір обумовлено певними об'єктивними характеристиками і властивостями даних алгоритмів.

Алгоритм **NSGA-II** має такі переваги:

- Зменшення складності алгоритму до $O(mN^3)$ шляхом використання оптимальних структур даних.
- Використання зовнішнього архіву з елітарними розв'язками.
- Реалізація розділення придатності без використання додаткових параметрів.
- Підтримка різноманіття в популяції.

Алгоритм **AMGA-2** має такі переваги:

- Залучення найоптимальніших практик.
- Збереження елітарних розв'язків.
- Збереження різноманіття.
- Мінімізація обчислень, робота з популяціями малого розміру.
- Можливість роботи майже з будь-яким типом кодування.
- Незначна чутливість алгоритму до зміни параметрів.

Алгоритм **ε-MOEA** має такі переваги:

- Підтримка добре розподілених розв'язків.
- Автоматичне обмеження результуючого архіву.

- Стійкість алгоритму.
- Стимуляція пошуку розв'язків, над якими не домінують інші.
- Підтримка різноманіття.
- Використання елітизму.

**Алгоритм NSGA-II**

В останні роки було запропоновано низку багатокритеріальних еволюційних алгоритмів. Головною причиною цього є їх здатність знаходити декілька оптимальних за Парето розв'язків за один прохід. Розглянемо детальніше багатокритеріальний еволюційний алгоритм на основі недомінантного сортування (Non-dominated Sorting GA-II або NSGA-II). По-перше, розглянемо підхід на основі швидкого недомінантного сортування, який характеризується складністю порядку $O(mN^2)$. По-друге, представимо оператор відбору, який створює пул схрещення, комбінуючи популяції батьків і нащадків і обираючи найкращі (залежно від придатності і розподіленості) N нащадків. Завдяки своїм незначним вимогам до обчислювальних ресурсів та використанню елітизму, і реалізації підходу розділення придатності без використання спеціального параметра, алгоритм NSGA-II знаходить широке застосування [17].

Генетичний алгоритм на основі недомінантного сортування (NSGA), запропонований в роботі [18], був одним з перших подібних еволюційних алгоритмів. Головні недоліки NSGA були такі:

- Висока обчислювальна складність недомінантного сортування. Складність алгоритму недомінантного сортування становила $O(mN^3)$, що в разі значних розмірів популяції означало підвищене споживання часу і обчислювальних ресурсів (члени популяції сортуються у кожному поколінні).
- Відсутність елітизму. Дослідження показують, що використання елітизму може значно прискорити роботу ГА, а також попередити втрату гарних розв'язків [19].
- Необхідність вводити параметр розділення придатності. Традиційні механізми підтримки різноманітності в популяції (для отримання широкого набору еквівалентних розв'язків) жорстко залежали від концепції розділення. Головною проблемою розділення придатності є необхідність специфікації параметра розділення придатності. Незважаючи на те, що робились спроби динамічного розрахунку розміру параметра розділення, збереження різноманіття без використання параметра є бажаним.

Значно вдосконалена версія NSGA, яка справляється з описаними вище недоліками, називається NSGA-II. Аналізуючи результати моделювання на наборах простих тестів, доходимо до висновку, що NSGA-II надає кращий набір оптимальних розв'язків у порівнянні. Наведено головні модулі, які формують NSGA-II.

*Модуль недомінантного сортування*. Для того, щоб відсортувати популяцію розміру N відповідно до рівня недомінантності, кожен розв'язок необхідно порівняти з усіма іншими розв'язками популяції для того, щоб визначити, чи домінує хтось із них над ним. Це вимагає O(mN) порівнянь для кожного розв'язку, де m – це кількість цілей. Коли цей процес виконується для знаходження членів першого недомінантного класу для всіх членів популяції, загальна складність становить $O(mN^2)$. На цьому етапі знаходимо усіх індивідів з першого недомінантного фронту. Для того, щоб знайти індивідів наступного фронту, розв'язки першого фронту вилучаються з розгляду і знову виконується описана вище процедура. У найгіршому випадку, задача знаходження другого фронту також вимагає розрахунків складністю $O(mN^2)$. Процедура повторюється для знаходження усіх інших фронтів. Очевидним є те, що у найгіршому випадку (коли на кожному фронті міститься лише один розв'язок) складність алгоритму дорівнюватиме $O(mN^3)$. Опишемо підхід на основі швидкого недомінантного сортування, який у гіршому випадку потребуватиме $O(mN^2)$ операцій.

По-перше, для кожного розв'язку обраховуватимуть дві сутності:

1) $n_i$, кількість розв'язків, які домінують над розв'язком *i;*
2) $S_i$, набір розв'язків, над якими домінує розв'язок *i*.

Розрахунок цих двох значень потребуватиме $O(mN^2)$ порівнянь. Ми визначаємо усі точки, для яких $n_i = 0$ і розміщуємо їх у списку F1. Будемо називати F1 поточним фронтом. Тепер, для кожного

розв'язку з поточного фронту ми дістаємо кожний елемент (*j*) з його множини $S_i$ і зменшуємо його $n_j$ на одиницю. При виконанні цієї процедури, якщо певний член *j* отримає нульове значення *n*, ми розміщуємо його в окремому списку H. Коли усі члени поточного фронту вже перевірені, ми визначаємо членів списку F1 як членів першого фронту. Після цього ми продовжуємо цей процес, використовуючи отриманий фронт H як поточний фронт.

Кожна така ітерація потребуватиме O(N) обчислень. Цей процес продовжується аж поки не будуть знайдені усі фронти. З тієї причини, що максимальне значення кількості фронтів – N, у найгіршому випадку складність цього циклу становитиме $O(N^2)$. Загальна складність алгоритму визначатиметься тепер як $O(mN^2) + O(N^2)$ або $O(mN^2)$.

Варто зазначити, що хоча обчислювальне навантаження зменшилося від $O(mN^3)$ до $O(mN^2)$ за рахунок підтримки допоміжних списків, обсяг споживання пом'яті прогнозовано збільшився від O(N) до $O(N^2)$ у найгіршому випадку.

Процедура швидкого недомінантного сортування під час застосування до популяції P повертає список недомінантних фронтів (F).

Алгоритм швидкого недомінантного сортування:

```
Для кожного p∈P
   Для кожного q∈P
      Якщо (p домінує над q) тоді
         Додати q до Sp
      Інакше якщо (q домінує над p) тоді
         Збільшити np на одиницю
   Якщо жоден розв'язок не домінує над p тоді
      Зробити p членом першого фронту
i = 1
Поки Fi≠∅ та H=∅
   Для кожного члена p Fi
      Кожний елемент q з множини Sp
         nq = nq – 1
         Якщо (nq = 0) тоді додати q до H
   i = i + 1
   Fi = H (сформувати поточний фронт з членів H)
```

*Оцінка щільності і оператор групового порівняння.* Для того, щоб отримати оцінку щільності розташування розв'язків навколо певної точки у популяції ми беремо середню відстань між двома точками по обидві сторони від точки вздовж кожної мети. Міра потрібна, як оцінка розміру найбільшого кубоіда, який обмежує точку і без включення будь-якої іншої точки з популяції (називатимемо це груповою відстанню). Для розрахунку групової відстані для кожної точки множини X використовується такий алгоритм:

```
Привласнення групової відстані(X)
Обрахувати кількість розв'язків у X і записати до l
Для кожного i, ініціалізувати значення відстані нулем.
Для кожної мети m
   Відсортувати множину X за значенням кожної функції мети
   Встановити для першого і останнього розв'язків максимально можливу
відстань (це забезпечить включення граничних значень)
   Для i від 2 до (l-1), тобто для усіх інших точок
      X[i]distance = X[i]distance + (X[i+1].m – X[i-1].m).
```

Тут X[i].m представляє значення m-ї функції мети для і-го індивіда у множині X. Складність цієї процедури обумовлюється алгоритмом сортування. У найгіршому випадку (коли кожен розв'язок на окремому фронті) сортування вимагатиме O(mNlogN) операцій.

Оператор групового порівняння ($\geq_n$) скеровує процес відбору на різних етапах роботи алгоритму у напрямку рівномірно розподіленого фронту оптимальних за Парето рішень. Припустимо, що кожний індивід і у популяції має два атрибути: ранг, та локальна групова відстань. Серед двох розв'язків з різними рангами домінування ми надаємо перевагу точці з нижчим рангом. У іншому випадку, якщо точки належать до одного фронту ми надаємо перевагу точці, яка розташована у регіоні з меншою кількістю точок (тобто коли розміри обмежуючого кубоїда більші).

***Головний цикл.*** На початку випадково створюється батьківська популяція $P_0$. Популяція сортується на основі інформації про домінування. Кожному розв'язку привласнюється значення придатності, яке дорівнює його рівню домінування (1 – це найкращий рівень). Отже, придатність підлягатиме мінімізації. Оператори відбору на основі бінарного турніру, рекомбінації, мутації використовуються для утворення популяції нащадків. $Q_0$ обсягу N. Починаючи з першого покоління, процедура набуває іншого вигляду. Процедура з урахуванням елітизму для $t \geq 1$ і для певного покоління має такий вигляд:

```
R_t = P_t ∪ Q_t (комбінувати популяції батьків і нащадків)
F = швидке недомінантне сортування (R_t).
Поки |P_{t+1}| < N (поки формування батьківської популяції не завершено)
   Привласнення групової відстані (Fi)
   P_{t+1} = P_{t+1} ∪ F_i (додати і-ий фронт до батьківської популяції)
Сортувати у порядку зменшення P_{t+1} використовуючи оператор ≥_n
P_{t+1}=P_{t+1}[0:N] (обрати перші N елементів з P_{t+1})
Q_{t+1} = нова популяція(P_{t+1}) (утворити нову популяцію шляхом використання
операторів селекції, кросоверу та мутації)
t = t + 1
```

На початку формується комбінована популяція $R_t = P_t \cup Q_t$. Розмір популяції $R_t$ становитиме 2N. Після цього популяція $R_t$ сортується, враховуючи інформацію про домінування. Нова батьківська популяція $P_{t+1}$ формується додаванням розв'язків з першого фронту поки її розмір не перевищить N. Отже, розв'язки з останнього отриманого фронту сортуються з використанням оператора $\geq_n$ і після цього обираються перші N точок. Саме так утворюється популяція $P_{t+1}$ розміром N. Ця популяція розміром N далі використовується для селекції, кросоверу та мутації для створення нової популяції $Q_{t+1}$ розміром N. Важливо зазначити, що ми використовуємо оператор відбору за бінарним турніром, але критерій відбору використовує оператор порівняння $\geq_n$.

Розглянемо тепер оцінку складності однієї ітерації всього алгоритму. Базові операції у найгіршому випадку призведуть до такого стану:

1) недомінантне сортування – $O(mN^2)$;
2) привласнення групової відстані – O(mNlogN);
3) сортування з оператором $\geq_n$ - (2Nlog(2N)).

Отже, загальна складність описаного вище алгоритму становить $O(mN^2)$.

Різноманітність серед розв'язків, над якими не домінують інші розв'язки, забезпечується шляхом використання процедури групового порівняння, яка застосовується під час селекції або скорочення популяції. З тієї причини, що розв'язки змагаються за показником групової відстані, стає непотрібним використання додаткового параметра ніші (такого як $\sigma_{share}$ в алгоритмі NSGA). Незважаючи на те, що групова відстань обраховується в області значень функції мети, вона також може бути реалізована в області параметрів, якщо це є необхідним.

Отже, NSGA-II швидкий в плані обчислення еволюційний багатокритеріальний алгоритм на основі підходу недомінантного сортування. Успішно справляється з задачею збереження протяжності фронту розв'язків, над якими не домінують інші розв'язки. Беручи до уваги такі характеристики алгоритму, як стратегія на основі елітизму і відсутність вимог до спеціальних параметрів, можна стверджувати про доцільність широкого використання NSGA-II у практичних задачах.

**Алгоритм AMGA-2**

Запропонований алгоритм передбачає новий тип процедури відбору, яка оптимально використовує історію пошуку алгоритму і намагається мінімізувати кількість обчислень значення функції, необхідної для досягнення бажаного сходження. Запропонований алгоритм працює з популяціями дуже малого розміру і підтримує архів найкращих і різноманітних розв'язків для того, щоб після закінчення симуляції вивести велику кількість розв'язків, над якими не домінують інші розв'язки. Запропоновано також покращене формулювання для деяких існуючих технік збереження різноманіття. Необхідно розглянути певні аспекти реалізації, які сприяють покращенню ефективності алгоритму. Різностороннє випробування і порівняння запропонованого алгоритму з іншими сучасними алгоритмами демонструє його покращені можливості пошуку [20].

Алгоритм AMGA-2 є еволюційним алгоритмом оптимізації і використовує оператори генетичної варіації для породження нових розв'язків. Схема поколінь, яка була впроваджена у запропонованому алгоритмі, може бути класифікована як така з тієї причини, що під час певної ітерації (покоління), у процесі відбору (селекції) беруть участь тільки ті розв'язки, які були створені до поточної ітерації. Проте алгоритм генерує невелику кількість нових розв'язків на кожній ітерації, отже, він може бути класифікований як майже стійкий генетичний алгоритм. Алгоритм працює з популяціями невеликого розміру і підтримує зовнішній архів отриманих непоганих розв'язків. На кожній ітерації, невелика кількість розв'язків створюється за допомогою операторів генетичної варіації. Новостворені розв'язки потім використовуються для оновлення архіву. Цей алгоритм отримав назву мікро-генетичний алгоритм з архівом (Archive-based Micro Genetic Algorithm - AMGA), враховуючи те, що він працює з популяціями дуже малого розміру і використовує зовнішній архів для підтримки його історії пошуку. Рекомендується використовувати архів значного обсягу для отримання великої кількості розв'язків, над якими не домінує жодний інший розв'язок. Розмір архіву визначає обчислювальну складність запропонованого алгоритму, хоча якщо розглядати задачі оптимізації, які потребують значних обчислень, час виконання алгоритму буде дуже малим порівняно з часом, який буде витрачено на підпрограми аналізу. Батьківська популяція створюється з архіву і відбір на основі бінарного турніру виконується для батьківської популяції для створення популяції нащадків. Алгоритм було розроблено так, що він не залежить від кодування змінних, саме тому запропонований алгоритм може працювати майже з будь-яким типом кодування (за умови надання відповідних операторів генетичної варіації). Алгоритм використовує концепцію призначення рангів за Парето, яка була запозичена з NSGA-II і базується на механізмі двошарової придатності. Псевдокод для запропонованого алгоритму (AMGA) можна подати так:

```
 1. Початок.
 2. Згенерувати початкову популяцію.
 3. Оцінити початкову популяцію..
 4. Оновити архів (використовуючи початкову популяцію).
 5. Виконати наступну послідовність дій.
 6. Створити батьківську популяцію на основі архіву.
 7. Підготувати індивіди до схрещення.
 8. Отримати популяцію нащадків.
 9. Оцінити популяцію нащадків.
10. Оновити архів (використовуючи популяцію нащадків).
11. Якщо не виконується умова завершення, перейти до 5.
12. Створити звіт на основі бажаної кількості розв'язків з архіву.
13. Кінець.
```

Хоча описаний вище псевдокод для AMGA є дуже простим, він чітко розділяє концептуальні кроки алгоритму. Цей псевдокод відображає роботу таких алгоритмів, як NSGA-II та SPEA2, адже їх поведінка цілком йому підпорядковується. Батьківська популяція утворюється з архіву шляхом застосування стратегії подібно до відбору з середовища, який використовується, наприклад, у SPEA2. Створення пулу схрещення базується на селекції шляхом бінарного турніру і схоже на той підхід, що використовується у NSGA-II. Будь-який оператор генетичної варіації може бути використаний для створення популяції нащадків. Стратегія, яка використовується для оновлення елітарної популяції (архіву), спирається на рівень домінування розв'язків, різноманіття розв'язків і поточний розмір архіву і базується на концепції недомінантного сортування, яке було запозичене з NSGA-II. Для того, щоб зменшити кількість обчислень функції для кожного покоління, AMGA використовує батьківські популяції і пул схрещення невеликих розмірів. Батьківська популяція створюється з архіву, використовуючи лише інформацію про різноманітність генотипів (змінних). Використання зовнішнього архіву, який зберігає велику кількість розв'язків, надає важливу інформацію про область пошуку, а також сприяє генерації у кінці симуляції великої кількості точок, над якими не домінують інші розв'язки. Розглянемо тепер детальніше кожний крок псевдокоду AMGA.

Нехай розмір батьківської популяції дорівнює N, а розмір архіву дорівнює A. Розмір популяції схрещення дорівнюватиме N/2. Нехай загальна кількість обчислень значення функції дорівнює T, а кількість функцій мети дорівнює M. Крок 2 алгоритму AMGA потребує O(N) часу. Крок 3 AMGA також потребує O(N) часу. На кроці 4, початкова популяція копіюється до архіву, це можна виконати за час O(N). Кроки з 5 по 11 являють собою головний цикл AMGA. Цикл ітерацій споживатиме найбільшу кількість часу на етапі наповнення архіву розв'язками, які належать до першого рангу. У найгіршому випадку час для виконання оператору різноманіття алгоритму AMGA буде пропорційним $O(MA^2\log(A))$. Задіяна недомінантна процедура призначення рангів споживатиме час $O(MA^2)$ (хоча існує і швидший метод). Кількість виконаних ітерацій залежить від кількості обчислень значень функції, а також розміру батьківської популяції. Для T обчислень значень функції кількість поколінь в алгоритмі становитиме $2(T-N)/N$. Як правило, $N << T$, отже, вираз можна спростити до $2T/N$. Тому можна очікувати, що складність алгоритму в найгіршому випадку становитиме $O(TMA^2\log(A)/N)$.

Алгориртм AMGA має такі відмінності: Він розроблений для роботи з популяціями дуже малого розміру і підтримує зовнішній архів з отриманими добрими розв'язками. AMGA також максимально використовує ідеї з відомої літератури і реалізує декілька концепцій, запозичених з наявних алгоритмів. В алгоритмі запропоновано покращену формулу для техніки збереження різноманіття. AMGA був розроблений для того, щоб полегшити незалежне налаштування параметрів алгоритму, тому розмір початкової популяції, розмір архіву, розмір батьківської (робочої) популяції може окремо налаштовувати користувач. Такий дизайн дозволяє встановлювати невеликий розмір для батьківської популяції, що своєю чергою значно прискорює процес пошуку розв'язків. AMGA також намагається мінімізувати вплив варіації індексу розподілення для кросоверу і мутації на успішне виконання алгоритму. Порівняння роботи алгоритму AMGA з NSGA-II і FastPGA демонструє, що AMGA працює не гірше, а у деяких випадках навіть краще за інші. Дослідження і розробку AMGA можна розглядати як прагнення скомбінувати і покращити найкращі ознаки наявних алгоритмів і найкращі техніки в уніфікованій процедурі оптимізації. Двома керуючими принципами, що вплинули на розробку AMGA, є:

1) прагнення зменшити загальну кількість обчислень значень функцій для того ж рівня сходження;

2) намагання зменшити чутливість алгоритму до змін параметрів.

AMGA певною мірою дав змогу досягнути цих цілей.

**Алгоритм ε-MOEA**

Алгоритм ε-MOEA, це стійкий варіант MOEA, який був розроблений на основі концепції ε-домінування [9].Область пошуку розділяється на рівні частини (або гіперкуби) і різноманіття підтримується за рахунок забезпечення того, що у кожній комірці чи гіперкубі міститься лише один розв'язок. Хоча PAES та споріднені з ним алгоритми були розроблені за схожим принципом, ε-

домінування є узагальненішою концепцією. У запропонованому багатокритеріальному еволюційному алгоритмі одночасно розвиваються дві популяції: поточна популяція *P(t)* та архівна популяція *E(t)* (де t – лічильник ітерацій).

Цей багатокритеріальний ГА стартує з початковою популяцією *P(0).* До архівної популяції *E(0)* заносяться розв'язки *P(0),* над якими не домінують інші розв'язки за принципом ε-домінування. Потім два розв'язки, один з *P(0)* і один з *E(0),* обираються для схрещення. Для того, щоб обрати розв'язок з *P(t),* обираються два випадкові члени цієї популяції і для них виконується перевірка. Якщо один розв'язок домінує над іншим, його обирають і виконують з ним подальші процедури. У іншому випадку розв'язки не домінують один над одним, тому ми просто випадково обираємо один з них. Нехай *p* – обраний розв'язок. Для того, щоб обрати розв'язок *e* з *E(t),* необхідно застосувати одну із стратегій, які встановлюють певний зв'язок з обраним розв'язком p. Проте, тут ми просто обираємо випадковий розв'язок з *E(t).* Після цієї фази відбору, розв'язки *p* і *e* схрещуються і утворюють λ розв'язків-нащадків ( $c_i, i = 1,2,...,\lambda$ ).

Для залучення розв'язку до архіву, нащадок $c_i$ порівнюється з кожним членом архіву і перевіряється на ε-домінування. Кожному розв'язку з архіву присвоюється визначальний масив ( $B = (B_1, B_2,..., B_M)^T$ , де M – це загальна кількість цілей), елементи якого розраховують так:

$$Bj(f) = \begin{cases} \lfloor (f_j - f_j^{\min}) / \varepsilon_j \rfloor, (\min(f_j)) \\ \lceil (f_j - f_j^{\min}) / \varepsilon_j \rceil, (\max(f_j)) \end{cases} \quad (1)$$

де $f_j^{\min}$ – це мінімальне можливе значення j-ї функції мети, а $\varepsilon_j$ – дозволений допуск для j-ї функції мети, нижче від якого два значення вважаються неістотними для користувача. Значення $\varepsilon_j$ чисельно дорівнює ε, яке використовується для визначення ε-домінування. Визначальний масив розділяє всю область рішень на гіперкуби, кожен з яких має розмір $\varepsilon_j$ для j-ї функції мети

Псевдокод алгоритму:

```
1. Створити початкову популяцію P(0).
2. Розв'язки, над якими не домінують інші за ε-критерієм, з P(0)
   переносяться до архівної популяції E(0).
3. Обрати одну особину з P і одну особину з E.
4. Ці індивіди породжують нащадка c.
5. Створюється спеціальний вектор B, який містить перетворені
   значення функцій мети для c.
6. Спробувати додати c до архівної популяції E.
7. Перевірка на домінування виконується з використанням вектора B, а
   не значень функцій мети.
8. Якщо c домінує на якимсь членом архіву, цей член заміщується c.
9. Особина c може також бути додана до P з використанням звичайної
   перевірки на домінування.
```

Описана вище процедура продовжується певну кількість ітерацій, і результуючі члени архіву повертаються як отримані розв'язки. Детальне обстеження цього алгоритму викриває такі його властивості:

1) він стійкий;

2) він стимулює пошук розв'язків, над якими не домінують інші;

3) він підтримує різноманіття в архіві, дозволяючи наявність лише одного розв'язку в кожному попередньо утвореному гіперкубі на фронті "Парето оптимальних" рішень.

**Реалізація алгоритмів NSGA-II, AMGA-2 та ε-MOEA**

Для тестового прикладу розглядалась задача керування розподілом ресурсів в енергосистемі. Обчислювалось споживання енергії в системі. Існують три типи ресурсів палива: побутовий газ, електричний струм та сонячна енергія. Крім цього, існують чотири типи енергетичних потреб: потреба в охолодженні, потреба в обігріванні, потреба в воді, і потреба в електричному струмі.

Цілями виступатимуть мінімізація витрат на купівлю обладнання, мінімізація витрат на установку обладнання, мінімізація споживання побутового газу та електричного струму для задоволення потреб. Кожна ціль знайшла своє відображення в якості функції мети. Крім цього, було введено необхідні обмеження. Одиниця вимірювання кіловат.

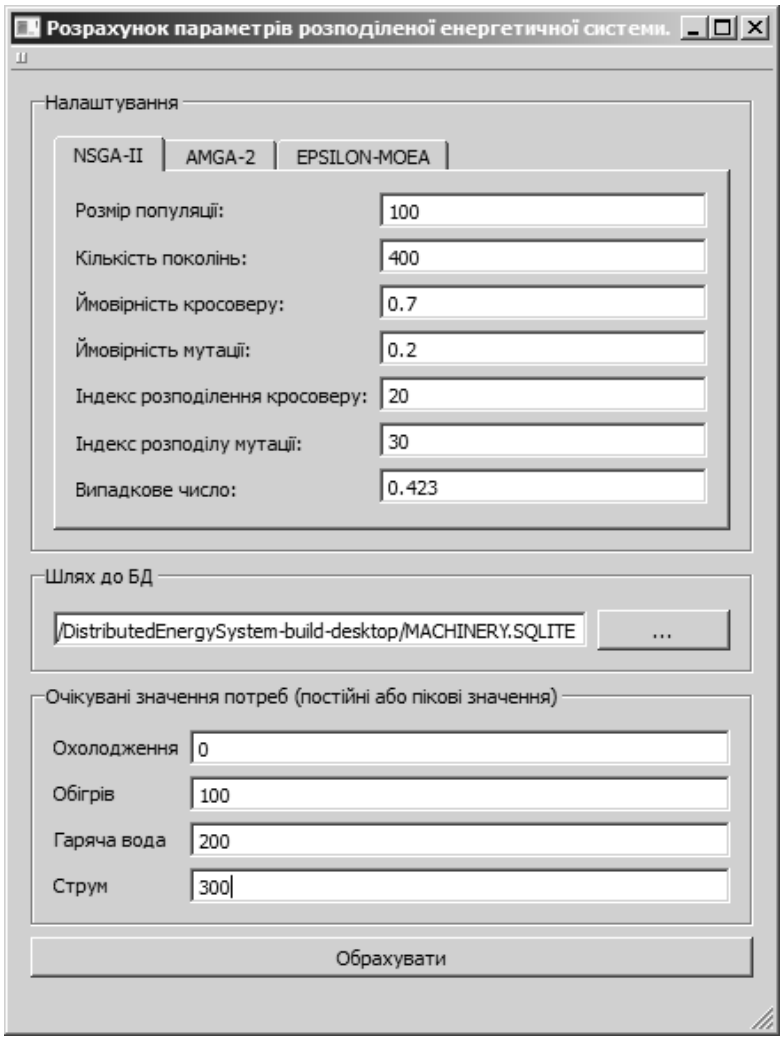


Рис. 1. Головна форма

Для параметрів алгоритмів за замовчанням і значень потреб (0; 100; 200; 300) отримали результати, зображені на рис. 2–5.

Результати виконання NSGA-II

| | ЕТН-1, кВт | АХМ, кВт | ЕТН-2, кВт | ГБ-1, кВт | ЕТН-3, кВт | ГБ-2, кВт | ТЕЦ, кВт | ФВ, кВт | спож. газу, кВт | спож. струму, кВт | поч. витрати, грн | вага, кг | площа, м^2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 180 | 0 | 0 | 83 | 23.3 | 600 | 5.2 | 253.43 | 29.6429 | 3.4186e+06 | 15786 | 552.462 |
| 2 | 0 | 0 | 0 | 0 | 118.2 | 0 | 700 | 0 | 330.556 | 42.2143 | 1.70017e+06 | 11880 | 19.9 |
| 3 | 0 | 180 | 0 | 0 | 118.2 | 0 | 600 | 5.2 | 251.852 | 42.2143 | 3.59152e+06 | 15891 | 552.37 |
| 4 | 0 | 180 | 0 | 0 | 83 | 23.3 | 600 | 0 | 253.43 | 29.6429 | 3.3583e+06 | 15775 | 212.462 |
| 5 | 0 | 180 | 0 | 0 | 118.2 | 0 | 600 | 0 | 251.852 | 42.2143 | 3.53122e+06 | 15880 | 212.37 |
| 6 | 0 | 0 | 0 | 0 | 83 | 23.3 | 700 | 0 | 332.133 | 29.6429 | 1.52725e+06 | 11775 | 19.992 |

*Рис. 2. Результати виконання NSGA-II*

Результати виконання AMGA-2

| | ЕТН-1, кВт | АХМ, кВт | ЕТН-2, кВт | ГБ-1, кВт | ЕТН-3, кВт | ГБ-2, кВт | ТЕЦ, кВт | ФВ, кВт | спож. газу, кВт | спож. струму, кВт | поч. витрати, грн | вага, кг | площа, м^2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 23.3 | 83 | 23.3 | 700 | 0 | 333.711 | 29.6429 | 1.53164e+06 | 11800 | 20.084 |
| 2 | 0 | 0 | 0 | 31.3 | 118.2 | 0 | 600 | 0 | 253.998 | 43.0851 | 1.60614e+06 | 11810 | 14.52 |
| 3 | 0 | 0 | 0 | 23.3 | 118.2 | 0 | 600 | 5.2 | 253.43 | 42.2143 | 1.66486e+06 | 11816 | 354.492 |
| 4 | 0 | 0 | 0 | 23.3 | 48.3 | 23.3 | 1000 | 0 | 462.615 | 17.25 | 1.69821e+06 | 11930 | 49.184 |

*Рис. 3. Результати виконання AMGA-2*

Результати виконання EPSILON-MOEA

| | ЕТН-1, кВт | АХМ, кВт | ЕТН-2, кВт | ГБ-1, кВт | ЕТН-3, кВт | ГБ-2, кВт | ТЕЦ, кВт | ФВ, кВт | спож. газу, кВт | спож. струму, кВт | поч. витрати, грн | вага, кг | площа, м^2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 48.3 | 180 | 0 | 23.3 | 83 | 23.3 | 600 | 5.2 | 255.008 | 46.8929 | 3.63242e+06 | 17291 | 561.554 |
| 2 | 48.3 | 180 | 0 | 23.3 | 83 | 23.3 | 600 | 0 | 255.008 | 48.4785 | 3.57212e+06 | 17280 | 221.554 |

*Рис. 4. Результати виконання ε-MOEA*

Порівняльна таблиця

| | Назва | Час виконання, мс | Кількість рішень, штук | Кількість рішень, над якими не домінують інші, штук |
|---|---|---|---|---|
| 1 | NSGA-II | 7036 | 6 | 6 |
| 2 | AMGA-2 | 5613 | 4 | 3 |
| 3 | EPSILON-MOEA | 558 | 2 | 0 |

*Рис. 5. Порівняння результатів*

Для параметрів алгоритмів за замовчанням і значень потреб (200; 0; 100; 300) отримали результати, зображені на рис. 6–9.

Результати виконання NSGA-II

| | ЕТН-1, кВт | АХМ, кВт | ЕТН-2, кВт | ГБ-1, кВт | ЕТН-3, кВт | ГБ-2, кВт | ТЕЦ, кВт | ФВ, кВт | спож. газу, кВт | спож. струму, кВт | поч. витрати, грн | вага, кг | площа, м^2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 230 | 0 | 0 | 0 | 23.3 | 600 | 5.2 | 253.43 | 0 | 3.31976e+06 | 14436 | 543.462 |

*Рис. 6. Результати виконання NSGA-II*

Результати виконання AMGA-2

| | ЕТН-1, кВт | АХМ, кВт | ЕТН-2, кВт | ГБ-1, кВт | ЕТН-3, кВт | ГБ-2, кВт | ТЕЦ, кВт | ФВ, кВт | спож. газу, кВт | спож. струму, кВт | поч. витрати, грн | вага, кг | площа, м^2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 230 | 0 | 23.3 | 118.2 | 23.3 | 0 | 20 | 47.0648 | 322.214 | 3.01832e+06 | 6292 | 657.154 |
| 2 | 48.3 | 180 | 0 | 23.3 | 118.2 | 23.3 | 0 | 7.8 | 37.5194 | 351.664 | 2.76753e+06 | 7433 | 616.154 |
| 3 | 0 | 230 | 0 | 23.3 | 0 | 23.3 | 600 | 5.2 | 255.008 | 3.24348 | 3.32415e+06 | 14461 | 543.554 |
| 4 | 0 | 230 | 0 | 23.3 | 0 | 23.3 | 600 | 20 | 255.008 | 0 | 3.54615e+06 | 14512 | 653.554 |
| 5 | 0 | 230 | 0 | 23.3 | 0 | 23.3 | 700 | 5.2 | 333.711 | 0 | 3.42415e+06 | 14561 | 549.054 |
| 6 | 48.3 | 180 | 0 | 23.3 | 83 | 23.3 | 0 | 20 | 37.5194 | 326.893 | 2.79042e+06 | 7342 | 666.154 |
| 7 | 48.3 | 180 | 0 | 23.3 | 118.2 | 23.3 | 0 | 5.2 | 37.5194 | 354.264 | 2.74573e+06 | 7421 | 556.154 |
| 8 | 0 | 230 | 0 | 23.3 | 0 | 23.3 | 600 | 7.8 | 255.008 | 0.643481 | 3.34595e+06 | 14473 | 603.554 |

*Рис. 7. Результати виконання AMGA-2*

Результати виконання EPSILON-MOEA

| | ЕТН-1, кВт | АХМ, кВт | ЕТН-2, кВт | ГБ-1, кВт | ЕТН-3, кВт | ГБ-2, кВт | ТЕЦ, кВт | ФВ, кВт | спож. газу, кВт | спож. струму, кВт | поч. витрати, грн | вага, кг | площа, м^2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 48.3 | 180 | 0 | 23.3 | 48.3 | 0 | 600 | 0 | 253.43 | 36.0683 | 3.4183e+06 | 17085 | 221.462 |
| 2 | 48.3 | 180 | 0 | 23.3 | 48.3 | 0 | 600 | 5.2 | 253.43 | 34.5 | 3.4786e+06 | 17096 | 561.462 |

*Рис. 8. Результати виконання ε-MOEA*

Порівняльна таблиця

| | Назва | Час виконання, мс | Кількість рішень, штук | Кількість рішень, над якими не домінують інші, штук |
|---|---|---|---|---|
| 1 | NSGA-II | 5954 | 1 | 1 |
| 2 | AMGA-2 | 5532 | 8 | 4 |
| 3 | EPSILON-MOEA | 118 | 2 | 2 |

*Рис. 9. Порівняння результатів*

Для параметрів алгоритмів за замовчанням і значень потреб (0; 0; 300; 100) отримали результати, зображені на рис. 10–13.

Результати виконання NSGA-II

| | ЕТН-1, кВт | АХМ, кВт | ЕТН-2, кВт | ГБ-1, кВт | ЕТН-3, кВт | ГБ-2, кВт | ТЕЦ, кВт | ФВ, кВт | спож. газу, кВт | спож. струму, кВт | поч. витрати, грн | вага, кг | площа, м^2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 180 | 0 | 0 | 172.4 | 23.3 | 700 | 0 | 332.133 | 61.5714 | 4.11679e+06 | 16215 | 217.962 |

*Рис. 10. Результати виконання NSGA-II*

Результати виконання AMGA-2

| | ЕТН-1, кВт | АХМ, кВт | ЕТН-2, кВт | ГБ-1, кВт | ЕТН-3, кВт | ГБ-2, кВт | ТЕЦ, кВт | ФВ, кВт | спож. газу, кВт | спож. струму, кВт | поч. витрати, грн | вага, кг | площа, м^2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 172.4 | 37 | 600 | 5.2 | 254.318 | 61.5714 | 2.14702e+06 | 12028 | 354.54 |

*Рис. 11. Результати виконання AMGA-2*

Результати виконання EPSILON-MOEA

| | ЕТН-1, кВт | АХМ, кВт | ЕТН-2, кВт | ГБ-1, кВт | ЕТН-3, кВт | ГБ-2, кВт | ТЕЦ, кВт | ФВ, кВт | спож. газу, кВт | спож. струму, кВт | поч. витрати, грн | вага, кг | площа, м^2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 180 | 0 | 23.3 | 172.4 | 37 | 600 | 5.2 | 255.896 | 61.5714 | 4.08247e+06 | 16153 | 552.602 |

*Рис. 12. Результати виконання ε-MOEA*

Порівняльна таблиця

| | Назва | Час виконання, мс | Кількість рішень, штук | Кількість рішень, над якими не домінують інші, штук |
|---|---|---|---|---|
| 1 | NSGA-II | 5931 | 1 | 1 |
| 2 | AMGA-2 | 5836 | 1 | 1 |
| 3 | EPSILON-MOEA | 563 | 1 | 0 |

*Рис. 13. Порівняння результатів*

Аналіз результатів свідчить про те, що NSGA-II вимагає для роботи більше часу порівняно з іншими алгоритмами, але усі результуючі рішення є потенційно оптимальними. AMGA-2 вимагав трохи менше часу, але в результаті певна кількість розв'язків відсікається. Нарешті, ε-MOEA виявився найшвидшим, але прослідковується тенденція повертати розв'язки, над якими домінують рішення, отримані за допомогою інших алгоритмів.

Програмно-аналітичний комплекс для перевірки ефективності обраних генетичних алгоритмів розроблявся за допомогою Qt -кросплатформеного інструментарію розробки ПЗ мовою програмування C++. В середовищі розробки – Qt Creator. Qt Creator — середовище розробки, призначене для створення крос-платформових додатків з використанням бібліотеки Qt.

**Висновки**

Проаналізовані багатокритеріальні алгоритми дають змогу ефективно розв'язувати задачі багатокритеріальної оптимізації і прийняття рішень. Найефективнішим з проаналізованих алгоритмів визначено ε-MOEA.

*1. Субботін С. О., Олійник А. О., Олійник О. О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологіних і нейромережних моделей: Монографія. – Запоріжжя: ЗНТУ, 2009. – 375 с. 2. A. Konak, D. W. Coit, A. E. Smith. Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering and System Safety 91 (2006) 992-1007. 3. Holland J. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press; 1975. 4. Goldberg D. Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley; 1989. 5. Schaffer J. Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the international conference on genetic algorithm and their applications,1985. 6. E. Zitzler, L. Thiele. Multi-objective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans Evol Comput1999; 3(4): 257–71. 7. H. Lu, G. Yen. Rank-density-based multi-objective genetic algorithm and benchmark test function study. IEEE Trans Evol Comput 2003; 7(4): 325–43.*

*8. D. Goldberg, J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In: Genetic algorithms and their applications: proceedings of the second international conference on genetic algorithms, 28–31 July, 1987. Cambridge, MA, USA: Lawrence Erlbaum Associates; 1987. 9. C. Fonseca, P. Fleming. 10. Multiobjective genetic algorithms. In: IEE colloquium on 'Genetic Algorithms for Control Systems Engineering' (Digest No. 1993/130), 28 May 1993. London, UK: IEE; 1993. 11. E. Zitzler, M. Laumanns, L. Thiele. SPEA2: improving the strength Pareto evolutionary algorithm. Swiss Federal Institute Techonology: Zurich, Switzerland; 2001. 12. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 2002; 6(2): 182–97. 13. H. Lu, G. Yen. Rank-density-based multiobjective genetic algorithm and benchmark test function study. IEEE Trans Evol Comput 2003; 7(4): 325–43. 14. J. Morse. Reducing the size of the non-dominated set: pruning by clustering. Comput Oper Res 1980; 7(1–2): 55–66. 15. K. Deb, S. Agrawal, A. Pratap, T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. Kanpur genetic algorithms laboratory report number 200001: Indian institute of technology Kanpur. 16. N. Srinivas, and K. Deb. (1995) Multi-Objective function optimization using non-dominated sorting genetic algorithms, Evolutionary Computation, 2(3):221–248. 17. E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation. 18. E. Zitzler, and Thiele, L. (1998) Multiobjective optimization using evolutionary algorithms—A comparative case study. In Eiben, A. E., Back, T., Schoenauer, M., and Schwefel, H.-P., editors, Parallel Problem Solving from Nature, V, pages 292–301, Springer, Berlin, Germany. 19. J. Knowles, and D. Corne. (1999) The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation. Proceedings of the 1999 Congress on EvolutionaryComputation, Piscatway: New Jersey: IEEE Service Center, 98–105. 20. Rudolph, G. (1999) Evolutionary search under partially ordered sets. Technical Report No. CI-67/99, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany. 21. S. Tiwari, P. Koch, G. Fadel, K. Deb. AMGA: An archive-based micro genetic algorithm for multi-objective optimization. GECCO'08, July 12-16, 2008, Atlanta, Georgia, USA. 22. R. Ooka and G. Kayo, Development of Optimal Design Method for Distributed Energy System (Part. 3) Sensitivity Analysis with GA Parameters, SHASE Annual Meeting, 2008.*