ДОДАТОК А
КОД ВИКОРИСТАНИЙ У ПРОЕКТІ

А.1 Код для генерації тестової вибірки графів.

```
#include <fstream>
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <stdlib.h>
#include <string>
using namespace std;
int main() {
	for (int l = 1; l < 16; ++l) {
		for (int k = 0; k < 16; ++k) {
			cout << to_string(k) + '_' + to_string(l) + '\n';
			ofstream fout("D:\\source\\diplom\\graphs\\" +
to_string(k) + '_' + to_string(l));
			int ma = pow(2, k);
			int na = pow(2, l);
			fout << pow(2, k) << endl;
			for (int i = 0; i < pow(2, k); ++i) {
				for (int j = 0; j < pow(2, k); ++j) {
					fout << (rand() % na) * (rand() % 2) << '
';
				}
				fout << endl;
			}
			fout.close();
		}
	}
	return 0;
}
```

А.2 Код для генерації тестової вибірки параметрів.

```
#include <fstream>
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <stdlib.h>
#include <string>
using namespace std;
int main() {
	for (int l = 1; l < 16; ++l) {
		for (int k = 0; k < 16; ++k) {
			cout << to_string(k) + '_' + to_string(l) + '\n';
			ofstream fout("D:\\source\\diplom\\parameters\\" +
to_string(k) + '_' + to_string(l));
			int ma = pow(2, k);
```

```
                    fout << rand() % ma << " " << rand() % ma << " " <<
rand() % 80 + 20;
                    fout.close();
                }
            }
            return 0;
        }
```

А.3 Код скрипту генеруючого БД.

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET                  @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET                                      @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,
NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema aco_test
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema aco_test
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `aco_test` DEFAULT CHARACTER SET utf8
;
USE `aco_test` ;

-- -----------------------------------------------------
-- Table `aco_test`.`graph`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `aco_test`.`graph` (
  `idGraph` INT(11) NOT NULL AUTO_INCREMENT,
  `numNodes` INT(11) NOT NULL,
  `numEdges` INT(11) NOT NULL,
  `path` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idGraph`))
ENGINE = InnoDB
```

```
AUTO_INCREMENT = 106
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `aco_test`.`parameters`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `aco_test`.`parameters` (
  `idParameters` INT(11) NOT NULL AUTO_INCREMENT,
  `start` INT(11) NOT NULL,
  `end` INT(11) NOT NULL,
  `iterations` INT(11) NULL DEFAULT NULL,
  `path` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idParameters`))
ENGINE = InnoDB
AUTO_INCREMENT = 259
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `aco_test`.`realization`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `aco_test`.`realization` (
  `idRealization` INT(11) NOT NULL AUTO_INCREMENT,
  `path` VARCHAR(45) NOT NULL,
  `numAnts` INT(11) NULL DEFAULT NULL,
  `vapCoef` FLOAT NULL,
  `isGpu` TINYINT(4) NOT NULL,
  PRIMARY KEY (`idRealization`))
ENGINE = InnoDB
AUTO_INCREMENT = 5
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `aco_test`.`experiment`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `aco_test`.`experiment` (
  `idExperiment` INT(11) NOT NULL AUTO_INCREMENT,
  `accuracy` FLOAT NOT NULL,
```

```
  `numRuns` INT(11) NOT NULL,
  `time` INT(11) NOT NULL,
  `isCrashed` TINYINT(4) NOT NULL,
  `Graph_idGraph` INT(11) NOT NULL,
  `Realization_idRealization` INT(11) NOT NULL,
  `Parameters_idParameters` INT(11) NOT NULL,
  PRIMARY    KEY    (`idExperiment`,    `Parameters_idParameters`,
`Realization_idRealization`, `Graph_idGraph`),
  INDEX `fk_Experiment_Graph_idx` (`Graph_idGraph` ASC) VISIBLE,
  INDEX                          `fk_Experiment_Realization1_idx`
(`Realization_idRealization` ASC) VISIBLE,
  INDEX  `fk_Experiment_Parameters_idx`  (`Parameters_idParameters`
ASC) VISIBLE,
  CONSTRAINT `fk_Experiment_Graph`
    FOREIGN KEY (`Graph_idGraph`)
    REFERENCES `aco_test`.`graph` (`idGraph`),
  CONSTRAINT `fk_Experiment_Parameters`
    FOREIGN KEY (`Parameters_idParameters`)
    REFERENCES `aco_test`.`parameters` (`idParameters`),
  CONSTRAINT `fk_Experiment_Realization1`
    FOREIGN KEY (`Realization_idRealization`)
    REFERENCES `aco_test`.`realization` (`idRealization`))
ENGINE = InnoDB
AUTO_INCREMENT = 2468
DEFAULT CHARACTER SET = utf8;

USE `aco_test` ;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

1. Код збереженної процедури add_experiment.

```
CREATE  DEFINER=`root`@`localhost`  PROCEDURE  `add_experiment`(IN
st INT, IN en INT, IN iter INT, IN parameters_path VARCHAR(45),
IN ants INT, IN vap INT, IN isgpu TINYINT, IN realization_path
VARCHAR(45),  IN  nodes  INT,  IN  edges  INT,  IN  graph_path
```

```
VARCHAR(45), IN accuracy FLOAT, IN numruns INT, IN speed INT, IN
crashed TINYINT, IN accuracy_abs FLOAT)
BEGIN
     IF ((SELECT COUNT(`path`) FROM `parameters` as a WHERE
a.`path` = parameters_path) = 0)
    THEN INSERT INTO `parameters` VALUES (NULL, st, en, iter,
parameters_path);
    END IF;
    IF ((SELECT COUNT(`path`) FROM `realization` as a WHERE
a.`path` = realization_path) = 0)
    THEN INSERT INTO `realization` VALUES (NULL,
realization_path, ants, vap, isgpu);
    END IF;
    IF ((SELECT COUNT(`path`) FROM `graph` as a WHERE a.`path` =
graph_path) = 0)
    THEN INSERT INTO `graph` VALUES (NULL, nodes, edges,
graph_path);
    END IF;
    INSERT INTO `experiment` VALUES(NULL, accuracy, numruns,
speed, crashed, (SELECT `idGraph` from `graph` where
`graph`.`path` = graph_path), (SELECT `idRealization` from
`realization` where `realization`.`path` = realization_path),
(SELECT `idParameters` from `parameters` where
`parameters`.`path` = parameters_path), accuracy_abs);
END
```

А.4 Код фрагментів обгортки

```
int query_state;
mysql_init(&mysql);
connection = mysql_real_connect(&mysql, "localhost", "root",
"Password98", "aco_test", 3306, 0, 0);
if (connection == NULL) {
     std::cout << mysql_error(&mysql) << std::endl;
     return 1;
}

query_state = mysql_query(connection, string("CALL
add_experiment(" + to_string(st) + ", " + to_string(en) + ", " +
to_string(iterations) + ", \"" + argv[2] + "\", " +
to_string(num_ants) + ", " + to_string(vap_coef) + ", 1, \"" +
```

```
argv[0] + "\", " + to_string(length) + ", " + to_string(get_edges()) + ", \"" + argv[1] + "\", " + to_string(1.0*correct / repeat) + ", " + to_string(repeat) + ", " + to_string(begin_clock2) + ", 0, " + to_string(acc/repeat) + ")").c_str());
if (query_state != 0) {
    std::cout << mysql_error(connection) << std::endl;
    return 1;
}
```

А.5 Код BAT-скріпту що запускає тестування

```
for /L %%f in (9,1,12) do (
    for /L %%d in (6,1,8) do (
        for /L %%i in (1,1,8) do (
            for /L %%j in (1,1,5) do (
                start /MIN /wait "D:\source\Diplom\Realizations\" 3.exe %%f_%%d %%i_%%j
            )
        )
    )
)
```

А.6 Код функції обробника помилок CUDA

```
template<typename T>
void check(T err, const char* const func, const char* const file, const int line) {
    if (err != cudaSuccess) {
        int state = mysql_query(connection, string("CALL add_experiment(" + to_string(st) + ", " + to_string(en) + ", " + to_string(iterations) + ", \"" + arg3 + "\", " + to_string(num_ants) + ", " + to_string(vap_coef) + ", 1, \"" + arg1 + "\", " + to_string(length) + ", " + to_string(get_edges()) + ", \"" + arg2 + "\", 0, " + to_string(repeat) + ", " + to_string((clock() - begin_clock)) + ", 1, 0)").c_str());
        std::cerr << "CUDA error at: " << file << ":" << line << std::endl;
        std::cerr << cudaGetErrorString(err) << " " << func << std::endl;
        exit(1);
    }
```

```
}
```

А.7 Код кернелу що ініціалізує curandState

```
__global__ void setup_kernel(curandState *state) {
     int idx = threadIdx.x + blockIdx.x * blockDim.x;
     curand_init(1234, idx, 0, &state[idx]);
}
```

А.8 Код SQL запиту для визначення неправильних тестів

```
select g.`path`, p.`path` from experiment as e, graph as g, parameters as p, realization as r where e.Realization_idRealization = 5 and e.Graph_idGraph between 45 and 66 and e.isCrashed = 0 and e.accuracy < 0.5 and e.`time` > 0 and g.idGraph = e.Graph_idGraph and p.idParameters = e.Parameters_idParameters and r.idRealization = e.Realization_idRealization
```

А.9 Код для генерації графів для якісного відображення швидкодії.

```
#include <fstream>
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <stdlib.h>
#include <string>
using namespace std;
int main() {
     for (int l = 8; l <= 1024; l += 8) {
          cout << to_string(l) + '\n';
          ofstream fout("D:\\source\\diplom\\graphs\\" +
to_string(l));
          fout << l << endl;
          for (int i = 0; i < l; ++i) {
               for (int j = 0; j < l; ++j) {
                    fout << (rand() % l) * (rand() % 2) << ' ';
               }
               fout << endl;
          }
          fout.close();
     }
     return 0;
}
```

А.10 Приклад коду для відображення графіку.

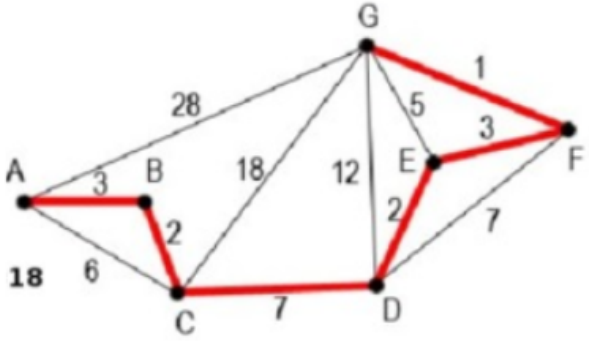
```
df[df.realization.isin([11,11])].groupby('num_nodes').mean().rena
me(columns = {"time":"vers 8, optimal"}).plot(y="vers 8,
optimal")
plt.show()
```

ДОДАТОК Б
ГРАФІЧНИЙ МАТЕРІАЛ

Задача пошуку мінімального шляху

Використання: картографія, радіоелектроніка, комп'ютерні мережі, data science, послідовність дій недетермінованої машини.

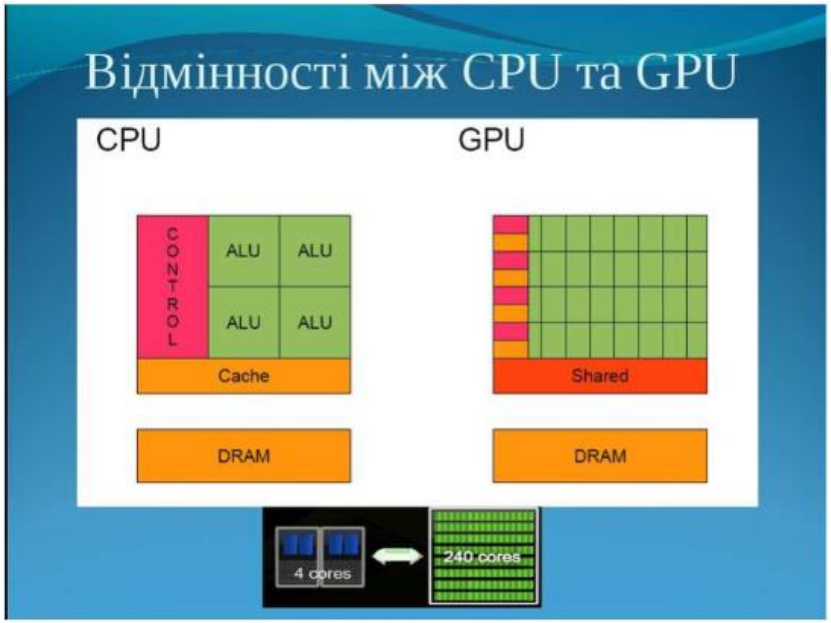
Проблеми: O(V^2) на щільних графах, нетривіальність паралельних реалізацій.



Розроб.	*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*
Перевір.				
Н. контр.				
				ХНУРЕ Кафедра СТ
Затв.				

General Purpose GPU

- GPU орієнтовані під массивні паралельні розрахунки;
- Багатоядерність + конвейерна архітектура, але менша кількість процесорних команд;
- Працює за принципами SIMD;
- До 10^5 резидентних потоків у картках останнього покоління.



Розроб.	*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*
Перевір.				
Н. контр.				
				ХНУРЕ
Затв.				*Кафедра СТ*

Математична постановка задачі. Методи вирішення.

Є граф G(V,E), де V - множина вершин, E - множина ребер.
Шлях у графі - послідовність вершин $P = (v_1, v_2, \ldots, v_n) \in V \times V \times \ldots \times V$
де між усіма v_i та v_{i+1} для $1 \leq i < n$ існує ребро. Нехай $e_{i,j}$ ребро що з'єднує v_i та v_j.
дана функція f($e_{i,j}$), що визначає вагу ребра. Тоді мінімальний шлях між вершинами v_i та v_j.
,$P = (v_1, v_2, \ldots, v_n)$ де $v_1 = v_j . v_n = v_i$ та $\sum_{i=1}^{n-1} f(e_{i,i+1})$ - мінімальна.

Алгоритм Дейкстри - O(V^2 + E);
Алгоритм Беллмана-Форда - O(V*E);
Алгоритм A* - O(V^2 + E);
Хвильовий алгоритм - O(E).

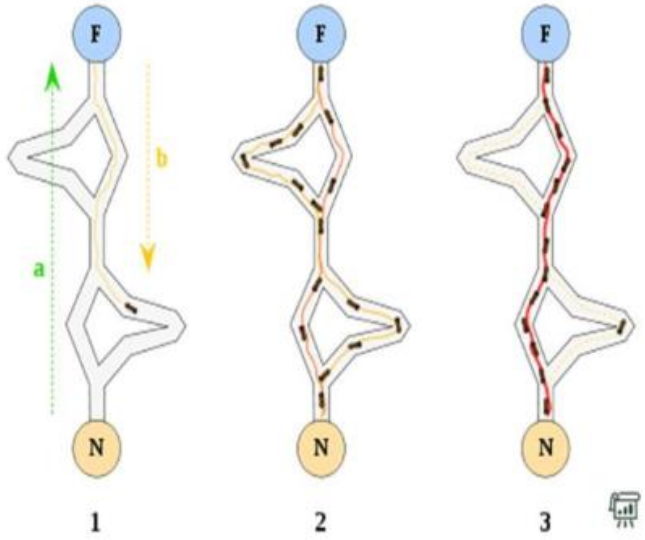
Розроб.	*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*
Перевір.				
Н. контр.				
				ХНУРЕ
Затв.				*Кафедра СТ*

Алгоритм Мурашиної Колонії

Суть: на ребрах є феромон, велика кількість мурах незалежно будують шляхи, обираючи ребра для переходу відносно кількості феромону, коли шляхи побудовані - оновлюється значення зворотньо пропорційно довжині шляхів. Таким чином, на певній ітерації отримаємо 1 шлях.

Особливості:

- Структурно відповідає принципам GPU;
- Може працювати за O(VlogV) для довільного графу;
- Великий простір до оптимізацій та модифікацій.



Розроб.	*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*
Перевір.				
Н. контр.				
				ХНУРЕ
Затв.				*Кафедра СТ*

Хвильовий алгоритм.

Суть: розповсюджуємо хвилю від початкової точки у всіх напрямках, на кожному кроці розповсюдження - збільшуючи значення на 1. На певній ітерації досягаємо кінецової точки.

Особливості:

- Працює лише з планарними графами з ребрами вагою 1;
- Будь який планарний граф може бути приведений до такого вигляду;
- Чітко зафіксовані точки взаємодії.

9	10		10	9	8	9	10	11	12	13	14
8	9		9	8	7	8	9	10	11	12	13
7	8	9	8	7	6	7	8	9	10	11	12
6	7	8	7	6	5	6	7			10	11
5					4	5	6	7	8	9	10
4	3	2	1	2	3	4	5	6			11
3	2	1	0	1	2	3	4	5			10
4	3	2	1	2	3	4	5	6	7	8	9

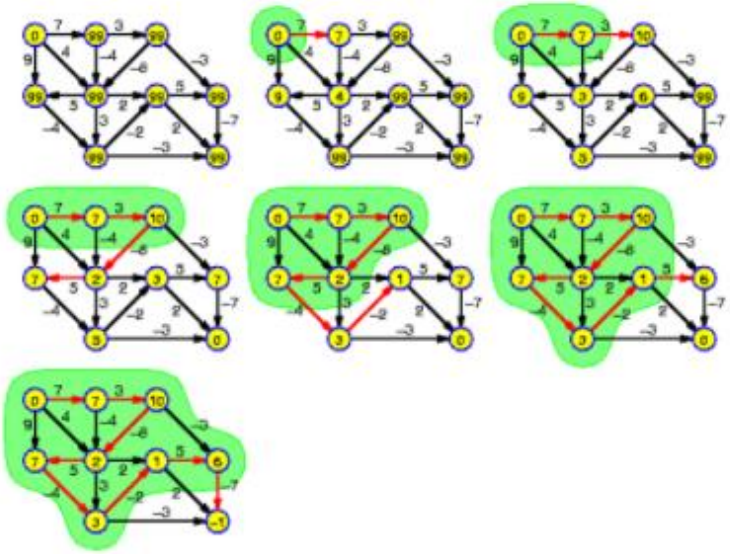
Розроб.	*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*
Перевір.				
Н. контр.				
				ХНУРЕ
Затв.				*Кафедра СТ*

Алгоритм Беллмана-Форда

Суть: у циклі оновлюємо значення мінімального шляху на вершинах, порівнюючи актуальне значення з сумою можливих переходів.

Особливості:

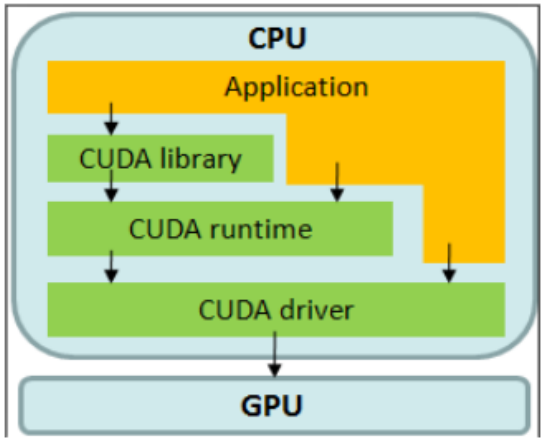
- Заснований на BFS;
- Може працювати для довільного графу;
- Не очевидна ідея паралельної реалізації.

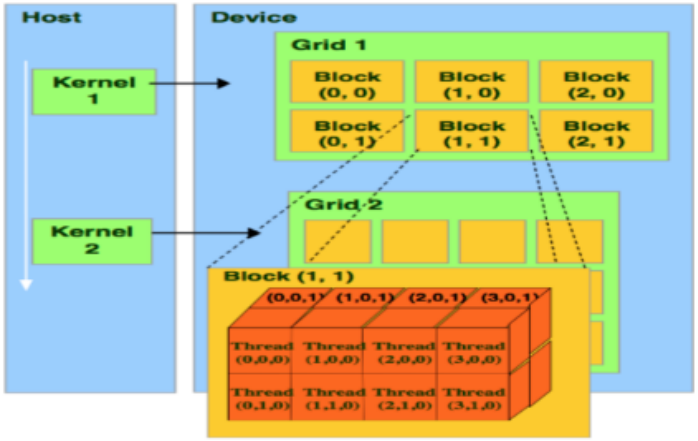


Розроб.	*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*
Перевір.				
Н. контр.				
				ХНУРЕ
Затв.				*Кафедра СТ*

GPGPU CUDA

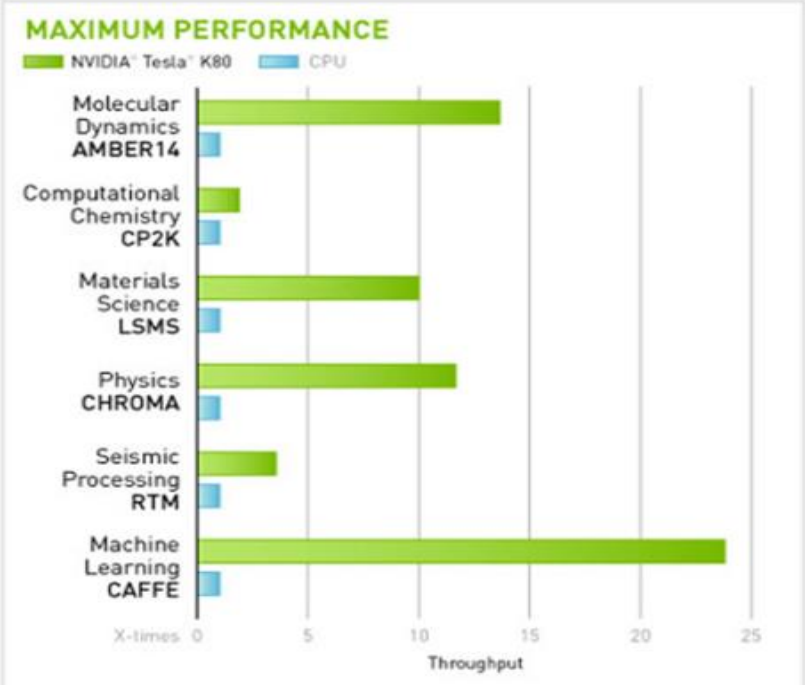
- 3-х вимірна, 3-х рівнева структура
- чіткі обмеження специфікацією
- 7 типів пам’яті

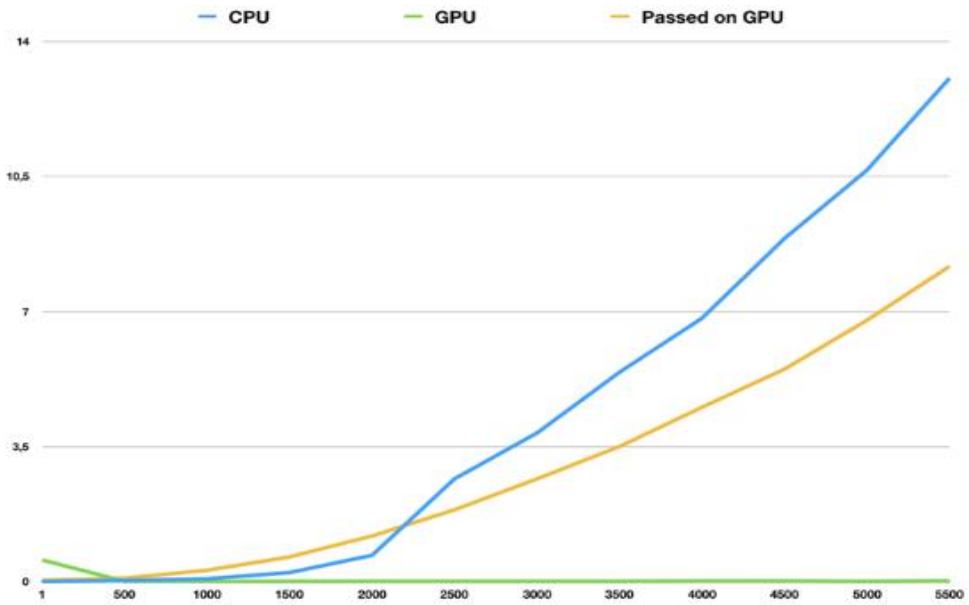




Розроб.	*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*
Перевір.				
Н. контр.				
				ХНУРЕ
Затв.				*Кафедра СТ*

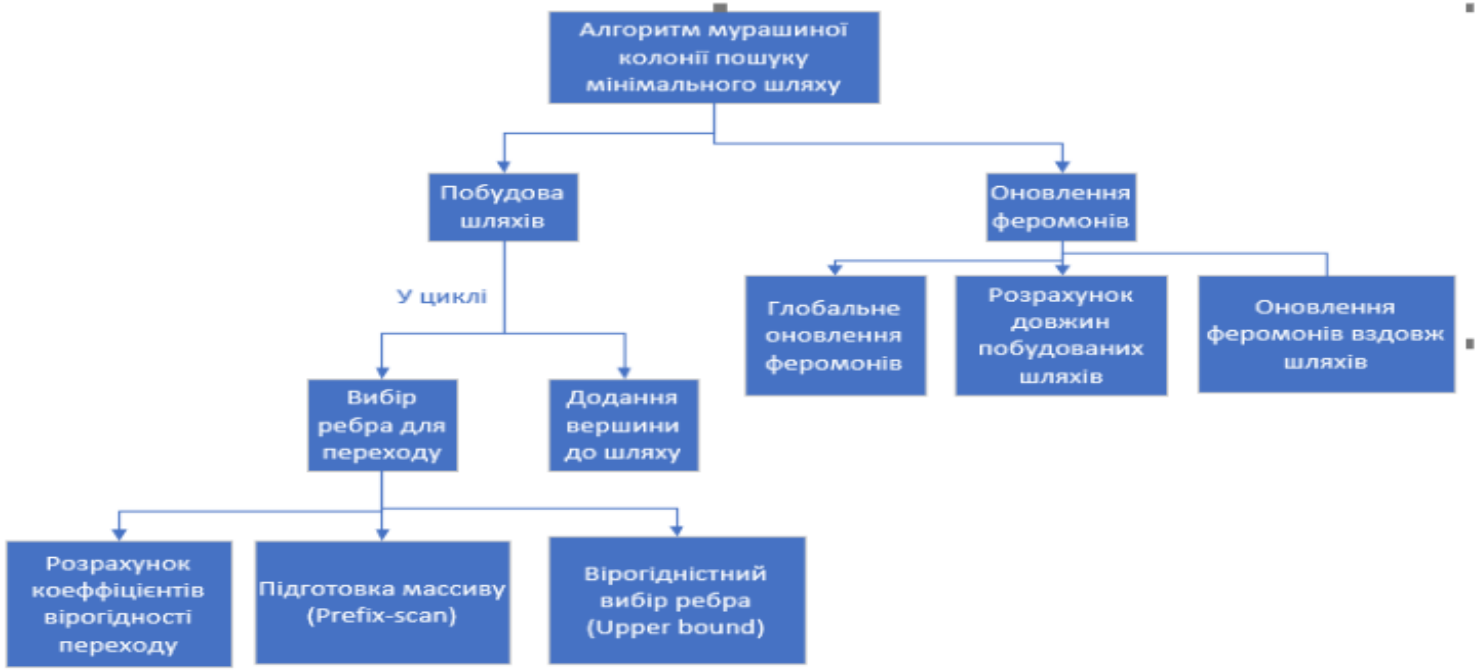
Порівняння швидкодії GPU та CPU





Розроб.	*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*
Перевір.				
Н. контр.				
				ХНУРЕ
Затв.				*Кафедра СТ*

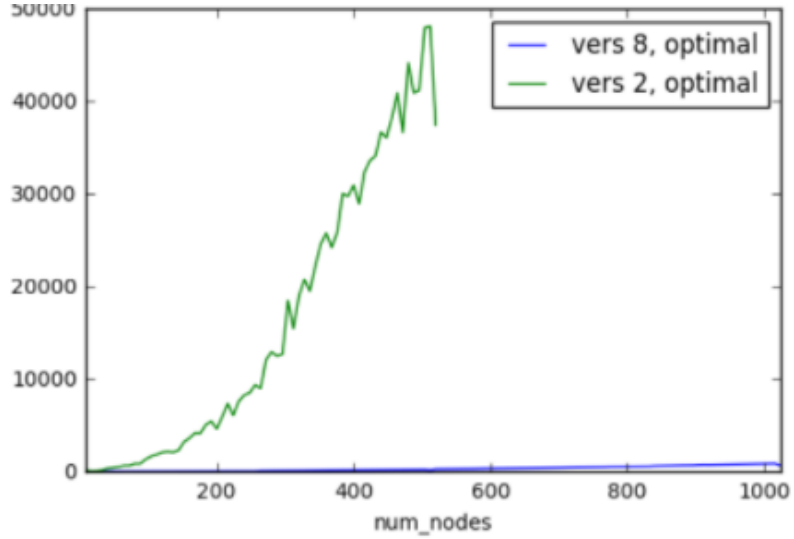
Структурна схема паралельних реалізацій алгоритму Мурашиної Колонії



Розроб.	*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*
Перевір.				
Н. контр.				
				ХНУРЕ
Затв.				*Кафедра СТ*

Основні віхи розробки алгоритму Мурашиної Колонії на GPU

1. Мураха це… (поток, блок потоків)
2. Субалгоритми, (sum, prefix scan, upper bound)... (бібліотека, власні імплементації, ефективні імплементації)
3. Пам'ять… (все в глобальній, майже все в розподіленій)
4. Відновлення шляху… (на GPU, на CPU, комбіноване)



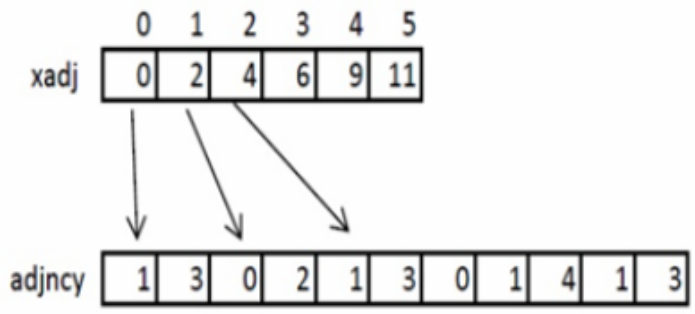
Розроб.	*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*
Перевір.				
Н. контр.				
				ХНУРЕ
Затв.				*Кафедра СТ*

Процесс розробки Хвильового алгоритму

1. Початкова ініціалізація (на CPU, на GPU)
2. Оновлення полів (всі, even/odd approach)
3. Умова виходу (V повторень, значення weight[end])

Процесс розробки Беллмана-Форда

1. Формат початкових даних (Adj, COO, CSR)



2. Оновлення полів (1 edge - 1 thread, 1 vertice - i thread)
3. Умова виходу (V повторень, окремий вектор changed, пошук його суми після кожної ітерації)

Розроб.	*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*
Перевір.				
Н. контр.				
				ХНУРЕ
Затв.				*Кафедра СТ*

Застосовані техніки

1. Представлення графу специфічною структурою
2. Розбиття even/odd
3. Використання CPU
4. Декомпозиція операцій
5. Мануальний контроль пам’яті
6. Заміна крокових алгоритмів на паралельно ефективні

Розроб.	*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*
Перевір.				
Н. контр.				
				ХНУРЕ
Затв.				*Кафедра СТ*

Ітогова ефективність

	Ефективність оригінального CPU алгоритму	Ефективність розробленого GPU алгоритму
Алгоритм Мурашиної Колонії	O(V^2*logV)	O(VlogV)
Хвильовий Алгоритм	O(V^2)	O(V)
Алгоритм Беллмана-Форда	O(V*E)	O(V^2)

Розроб.	*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*
Перевір.				
Н. контр.				
				ХНУРЕ
Затв.				*Кафедра СТ*

Публікації

Прийняття до публікації статті Дубинський, В.М, Лисенко, Б.С., Каяфюк, Р.В., Панкратов, О.В., Романова, Т.Є. Опукла багатокутна оболонка для пари складених об'єктів. Збірник "Кібернетика та комп'ютерні технології" № 2 за 2021 рік

Участь у 25-му міжнародному молодіжному форумі “РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ В XXI ст.” з доповіддю на тему “РОЗРОБКА ПІДСИСТЕМИ ДЛЯ РОЗВ’ЯЗКУ ЗАДАЧІ ПОШУКУ МІНІМАЛЬНОГО ШЛЯХУ З ВИКОРИСТАННЯМ GPU“.

№	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1.	ГЮИК.503000.009 ПЗ	Пояснювальна записка	99 стор.
		Графічні документи	
2.		Змістова постановка задачі	1 аркуш
		пошуку мінімального шляху	
3.		General Purpose GPU	1 аркуш
4.		Алгоритм мурашиної колонії	1 аркуш
5.		CUDA	1 аркуш
6.		Порівняння швидкодії GPU	1 аркуш
		та CPU	
7.		Хвильовий алгоритм	1 аркуш
8.		Алгоритм Беллмана-Форда	1 аркуш
9.		Структурна схема	1 аркуш
		паралельних реалізацій	
		алгоритму мурашиної колонії	
10.		Основні віхи розробки алгоритму	1 аркуш
		Мурашиної Колонії на GPU	
11.		Процес розробки Хвильового	1 аркуш
		алгоритму	
12.		Процес розробки Беллмана-Форда	1 аркуш
13.		Застосовані техніки	1 аркуш
14.		Ітогова ефективність	1 аркуш
15.		Публікації	1 аркуш

					ГЮИК.503000.009 ДЗ		
Змін.	Арк.	№ докум	Підп.	Дата			
Розроб.		*Каяфюк Р.В.*			*Дослідження методів пошуку оптимальних шляхів на графах з використанням графічного процесора*	Аркуш	Аркушів
Перевір.							
Т. Контр.							
Рецензн.						1	1
Н. Контр.		.				*ХНУРЕ Кафедра СТ*	
Затв.							