Факултет техничких наука

Универзитет у Новом Саду

Паралелне и дистрибуиране архитектуре и језици

Склапање слике употребом генетског алгоритма

Аутор:
Филип Стефанов

Индекс:
Е2 110/2023

31. јануар 2024.

Садржај

1 Преглед алгоритма **1**
1.1 Генетски алгоритам 1
1.2 A Genetic Algorithm-Based Solver for Very Large Jigsaw Puzzles 1

2 Имплементација **4**

1 Преглед алгоритма

Програм за склапање слике имплементиран у овом раду ослања се на истраживање Sholomon et al.[1] које је за циљ имало унапређење постојећих решења заснованих на генетском алгоритму. У наставку следи кратак опис основних идеја генетског алгоритма и његове употребе у домену проблема овог рада.

1.1 Генетски алгоритам

Генетски алгоритам представља оптимизациони алгоритам заснован на идејама природне селекције и генетике. Генетски алгоритам омогућава ефикасну претрагу простора решења за одређени проблем, симулирајући процесе еволутивног напретка и усавршавања врсте.

Полазећи од иницијалне популације јединки (јединке су кандидати за решење оптимизационог проблема), елитизмом и селективним укрштањем који фаворизују боље прилагођене јединке и мутацијама, које представљају случајне модификације, добија се нова популација сачињена од јединки код којих су задржане добре особине из претходне генерације. Свака нова генерација у просеку представља боље прилагођену популацију датим условима (садржи боља решења), чиме алгоритам конвергира ка глобалном оптимуму.

Прилагођеност јединке тј. решења (engl. fitness) представља критеријум тачности изражен помоћу математичке формуле. Циљ алгоритма јесте пронаћи јединку за коју се постиже глобално оптимална вредност прилагођености.

1.2 A Genetic Algorithm-Based Solver for Very Large Jigsaw Puzzles

Овај рад бави се истраживањем на тему примене генетског алгоритма за решавање пузли. Ослањајући се на постојеће радове и њиховим унапређењем, Sholomon et al. постигли су *state of the art* решење у домену склапања слика из великог броја делова.

Користећи терминологију из потпоглавља 1.1, у контексту склапања слика, јединку представља једно решење пузле, тј. делови слике сложени на одређени начин.

Прилагођеност јединке одређује се сумирањем различитости суседних ивица делова слике. Два суседна дела могу бити у четири просторне релације: *U - up* - изнад, *D - down* - испод, *R - right* - десно, *L - left* - лево.

Нпр. различитост ивица за део x_i који је у релацији R (десно) са делом x_j рачуна се по формули:

$$D(x_i, x_j, r) = \sqrt{\sum_{k=1}^{K}\sum_{b=1}^{3}(x_i(k, K, b) - x_j(k, 1, b))^2}$$

Користећи формулу за различитост, прилагођеност јединке рачуна се као:

$$\sum_{i=1}^{N}\sum_{j=1}^{M-1}(D(x_{i,j}, x_{i,j+1}, r)) + \sum_{i=1}^{N-1}\sum_{j=1}^{M}(D(x_{i,j}, x_{i+1,j}, d))$$

Из формуле за прилагођеност јасно се види да је нижа вредност последица мање различитости ивица у релацијама *D* и *R*. Циљ еволуције је доћи до јединке која има глобално минималну вредност за ову функцију.

Почевши од иницијалне популације, свака нова добија се коришћењем следећих операција: елитизам, укрштање и мутације.

Елитизам подразумева задржавање *n* најбоље прилагођених јединки из тренутне популације у следећој.

Мутације подразумевају насумичну измену јединке, чиме се уводе нове, потенцијално добре особине у новој популацији.

Приликом укрштања, насумично се бирају два родитеља методом *roulette wheel selection*, која фаворизује јединке са бољом прилагођеношћу. Укрштање се своди на генерисање нове јединке растућим кернелом који на почетку садржи један насумично одабрани део. Даље, кернелу се итеративно додају најкомпатибилнији слободни делови, и то на следећи начин:

1. прво се тражи део који је у оба родитеља у истој релацији у односу на део кернела *x* за који се тражи најкомпатибилнији слободан део

2. уколико не постоји слободан део за који се оба родитеља слажу, узима се *best buddy*. *Best buddies* представља пар делова који су један другом најсличнији у међусобно комплементарним релацијама. Формално, ово се изражава као:

$$\forall x_k \in Pieces,\ C(x_i, x_j, R_1) \geq C(x_i, x_k, R_1)$$
$$\text{and}$$
$$\forall x_p \in Pieces,\ C(x_j, x_i, R_2) \geq C(x_j, x_p, R_2)$$

3. Уколико је корак 2. неуспешан, додаје се најсличнији слободан део у траженој релацији

Овај поступак понавља се све док се не попуни слика, тј. док се не искористе сви делови, чиме је процес укрштања завршен.

Изгенерисана популација користи се у следећој итерацији за добијање нове популације. Критеријум заустављања јесте достизање максималног броја популација, који представља унапред задати параметар.

Слика испод приказује псеудокод генетског алгоритма.

```
1: population ← generate 1000 random chromosomes
2: for generation_number = 1 → 100 do
3:     evaluate all chromosomes using the fitness function
4:     new_population ← NULL
5:     copy 4 best chromosomes to new_population
6:     while size(new_population) ≤ 1000 do
7:         parent1 ← select chromosome
8:         parent2 ← select chromosome
9:         child ← crossover(parent1, parent2)
10:        add child to new_population
11:    end while
12:    population ← new_population
13: end for
```

2 Имплементација

Овај рад представља паралелну имплементацију програма за склапање слика у програмском језику Rust. Имплементација је заснована на раду [1].

Једина информација о оригиналној слици потребна за имплементирани алгоритам јесу димензије слике у пикселима. Када су познате, потребно је одредити број редова и колона дводимензионе матрице у коју се убацују делови слике, од које се на крају генерише сама слика. Ово се постиже дељењем димензија оригиналне слике просечном ширином и висином делова од којих се слика састављa. У циљу поједностављења алгоритма, сви делови своде се на просечне димензије. Ово не доводи до видљивих разлика између склопљене и оригиналне слике, иако се уводи релативно мала дисторзија на нивоу појединачних делова.

Након што су одређене димензије матрице, генеришу се речници за релације доле и десно. Ови речници као кључ садрже парове (x_i, x_j), док је вредност различитост ивица у датој релацији - $D(x_i, x_j, Rel)$. Итерације генерисања ових речника паралелизоване су помоћу паралелног итератора *Rayon create*-а. Ови речници служе као *lookup* табела, како би се убрзало рачунање прилагођености.

Поред овога, креира се и *Adjacency* структура, чији конструктор рачуна најсличније делове за сваки део у свакој релацији, као и *best buddy* парове. Сврха *Adjacency* структуре такође је убрзање програма. Одређивање најсличнијих делова паралелизовано је помоћу паралелног итератора *Rayon create*-а.

Након конструисања ових структура, генерише се иницијална популација са 500 јединки (број јединки по популацији је константан).

Затим, са извршавањем започиње *for* петља са унапред одређеним бројем итерација (у овом раду 30).

У оквиру сваке итерације прво се одређује прилагођеност свих јединки тренутне популације. Одређивање прилагођености врши се у паралели, коришћењем паралелног итератора из *Rayon create*-а.

Нова популација садржи 4 најелитнијих јединки из постојеће популације, док је остатак сачињен од јединки добијених укрштањем. С обзиром да су свака два укрштања међусобно независна, овај процес такође је паралелизован употребом паралелног итератора *Rayon create*-а.

За одабир родитеља коришћен је већ поменути *roulette wheel selection*. С обзиром на то да вероватноћа да јединка буде одабрана расте са њеном прилагођеношћу

која је изражена као сумирана различитост суседних делова (мања вредност значи боља прилагођеност), потребно је инвертовати ове вренодсти рачунањем њихове реципрочне вредности.

Након одређивања родитеља конструише се *Crossover* структура и покреће се генерисање детета на основу одабраних родитељских јединки. Ово представља итеративни процес који траје све док постоје неискоришћени кандидати за додавање у растући кернел. Нови кандидати одређују се након што се део дода у кернел. С обзиром на то да се делови додају на тренутне ивице кернела, након сваког додатог дела отварају се максимално 3 нова кандидатска места (слободна места у различитим релацијама у односу на последње додати део), чиме се тренутни кернел проширује, под условом да нису премашене димензије матрице слике. Кандидати за новоотворена места додају се поступком описаним у потпоглављу 1.2. За чување кандидата користи се *min-heap* у ком највиши приоритет имају кандидати за које се оба родитеља слажу, затим *best-buddies* и на крају најсличнији кандидати.

Процес укрштања успешно је завршен када су сви делови искоришћени тако да се на сваком месту матрице слике налази тачно један део.

Након генерисања популације започиње нова итерација у којој се врше евалуација и генерисање. Када се изврше све итерације, из последње генерисане популације узима се најбоље прилагођена јединка на основу које се формира слика, чиме је склапање завршено.

Библиографија

[1] A genetic algorithm-based solver for very large jigsaw puzzles. https://openaccess.thecvf.com/content_cvpr_2013/papers/Sholomon_A_Genetic_Algorithm-Based_2013_CVPR_paper.pdf.