SWARM PROJECT

Amélie NEF

6 June 2020

1 Context

ARGOS is a software to produce different simulation with a multi-physics robot. The aim of the this software is to efficiently practice SWARM optimization concepts(Carlo Pinciroli, 2012). In this project, the idea is to produce a solution with the coordination concept to solve the following problem: finding the best room with the highest purity value and aggregate the robots in this room. This purity value represents different attributes like:

1. **Floor color** (v_{Fi}): it's a palette of grey colors, with extremities [0:1]. 0 if the floor is black (RGB : 0,0,0) and 1 if is white (RGB : 1,1,1).
2. **Number of objects** (v_{Oi}): it's a simple detection of green LEDs which are present on each "object". The value of this component begins at 2 and goes up to 12.

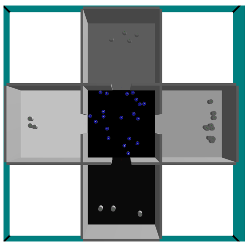
Figure 1: Screenshot of scenario

Formula $$v_i = \frac{v_{Fi}+v_{Oi}}{2}$$

A constraint to limit the memory will be applied to avoid the simple solution which consists in collecting all purity values of each room and then just go to the best room. We will only allow robots to remember only one purity value at any given time. The solution should take into account the opinions and information of other robots to try to choose the best room together to avoid errors and to converge faster.

To produce our solution, we will take into account **all notions discussed and proposed during the theoretical and practicals courses** of INFO H414 - Swarm Optimization. We did not look at the proposed extra literature for the project.

2 First step : produce a lua code to resolve the scenario

2.1 Initialization

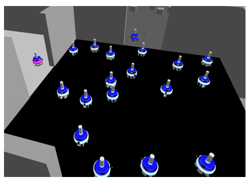
Figure 2: Screenshot of scenario

At the beginning, all robots are initialized with a specific beginning color (arbitrarily set to 128, 249, 255). This color marks robots that are "uninitialized" and have not yet visited any room. This is used during the votes which take place in the central room (this will be explained further the next section) to ignore robots that do not yet have any information to provide to the swarm. Each robot has 4 global variables which are:

PREVIOUS which represents the current known quality value for this robot, and is initialized to 0 (the worst possible purity value), and **R_ROOM_COLOR_ PREVIOUS, G_ROOM_COLOR_PREVIOUS, B_ROOM_ COLOR_PREVIOUS** which represent the RGB values of the best visited room, initialized with the RGB of the "uninitialized" color mentioned previously.

2.2 Strategies when a robot enters a candidate room

We use floor color to detect that a robot has entered a room. Any room with a floor different than RGB (0,0,0) is a candidate room (and not the central room).

2.2.1 Step 1 : Determine the quality value of the room

After entering a room, the robot computes the number of objects, and measures the floor RGB to produce the quality value of room according to the formula above.

2.2.2 Step 2 : Comparison of the current room quality with the value in the robot's memory

A comparison with the PREVIOUS value with the newly computed quality. The robot can face three possibilities:

Possibility 1 : it's the first room visited by the robot

A - When the robot enters in the room he changes his external color to take the color of the room which is designated by the LED present at the entry of the room. That could be help to speed taking the first group decision. This indicates the vote of the robot. He then randomly explores the room.

B - When the robot goes out the room : he updates the PREVIOUS value and the other global variables with the RGB of the visited room. To be able to distinguish between door LEDs and robot LEDs, we subtract 1 to each RGB component of a door LED when displaying it on a robot. This way, door LEDs will not interfere during the voting process.

The robot randomly explores the room in order to allow robots to find other robots and not get stuck on their first choice.

Possibility 2 : the current purity value is better or equal to the PREVIOUS encoded value

A - When the robot enters in the room the robot uses the new room color as his own displayed color, to signal his preference for this room. Similar as above.

B - During exploration of the room

B - 1 - Meeting with a robot with a different room color displayed with all his LED : if a robot meets another robot voting for a different color in this room, this means that this second robot has information about a better room. Otherwise, he would have changed his displayed color to the color of the color advertiser and move in the room to inform other robots that this is room is not the best. So the current robot will follow the vote of this second robot and change his displayed color accordingly. This new color prevent other robot to make an aggregate with this robot.

B - 2 - Meeting with a robot with current room color : If two robots are close and agree that the current room is the best, they stop their exploration and wait for other robots to gather in this "best" room. Since these robots may be both wrong, we allow them to "change their mind": with a probability of 0.035 (determined empirically) they can resume their search and look for another room. This probability is actually adjusted by the number of robots, to make it less likely to resume the search when the group of robots is large.

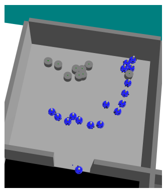
Figure 3: Screenshot of scenario

B - 3 - Absence of any meeting : the robot doesn't aggregate and continue his exploration of the room or moves in the central room.

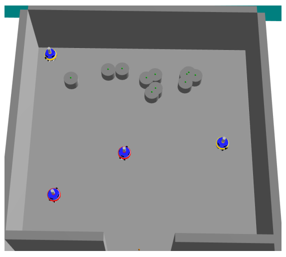

Figure 4: Screenshot of scenario

Possibility 3 : the actual purity value are lower than the PREVIOUS encoded value

A - when the robot is not too near to the door , he is going to propagate the information to the other robot presents in the room that is not the best room on the map. Doing so, the robot can force any incorrect aggregate of robots to resume their search.

2.3 Strategies when the robot is in the central room

If we assume as said that only the central room has an RGB value equal to (0,0,0), the robots are able to notice if they are in the central room or not with their own floor detector. The action specific to this central room will be a vote to determine collectively the best room and going to this room with an orientation movement.

This vote could help to speed up the execution and reach the goal faster (aggregating into the best room).

Before the vote, the robot has two possibilities:

Possibility 1 In this strategy, the robot will look at the result of the vote. This strategy has a 0.99 probability of being picked. The vote can have two outcomes:

A - The threshold (quorum) is not reached , in this case no collective decision could be made and the robot doesn't change his own external color and no does not move in particular direction.

B - The quorum is reached , in this case a collective decision can be made. At the end, the robot takes the winning color and makes a directional move to go to the door of the winning color.

Possibility 2 The robot can also choose to ignore the vote, and pick a random color then move towards this room. This strategy has a 0.01 chance of being selected.

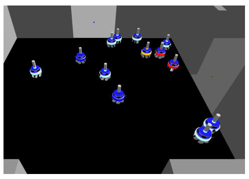

Figure 5: Screenshot of scenario

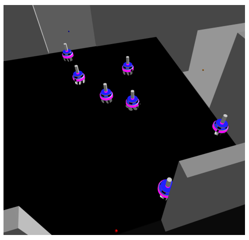

Figure 6: Screenshot of scenario

2.4 Notice

During the realisation of the second part project, we read on the Teams discussion that my assumption is not true (there may be other rooms with a black (0,0,0) floor). But taking into account the time to execute all the executions we needed, it was simply impossible to adapt the code and restart all processes. The modification could be to add a parameter in the first "if condition" like the distance between the robot and any object with a value of 300 to make sure we're in the central room.

3 Second step : optimize parameters with PSO optimization

3.1 PSO optimization

It's a meta-heuristic algorithm based on the concept of swarm intelligence capable of solving complex mathematics problems existing in engineering(de Almeida & Leite, May 30th 2019). He could be represent by a point (which are the particle in a X dimension world (X equals to the number of parameters that we want to optimize). The coordinates of particles are equals to a list of value of length equal to the number of parameters to explore. At the beginning, the coordinate are be randomly generate, the next step is the evaluation of the purity of this current position and move to take into account his own best position, the best position of his neighbors and finally adding a inertia parameter that could be a great role in the modulation of the equilibrium between exploration of the possibilities and exploitation of the data previously collect (INFOH414-Swarm optimization).

The aim of this project that could be provide a better parameters which we could put in the initial program into the idea of improve it. Before that we need to choose the parameters to optimize.

3.2 Determination of parameters to optimize

The goal in this subsection is to choose the parameters which are interesting to improve. We need to do that because it's kind impossible to improve all of them just by the criteria of the time we have.

3.2.1 List of all parameters used in the Lua program

1. **DISTANCE_MAXIMAL_ROBOT** : Distance between the robot and the closest robot displaying the current room color. It helps regulate the density of the aggregate.
2. **DISTANCE_MAXIMAL_OBJECT_STAY** : Distance between the robot and the furthest object of this room. Used in avoiding blocking an entry.
3. **DISTANCE_MAXIMAL_OBJECT_REBOOT** : Distance between the robot and the furthest object of this room. Used to configure the area where a reboot of robot color could be done.
4. **DISTANCE_MINIMAL_DOOR** : Distance between the robot and the entry. It is used to avoid a neverending reboot of the robot's color.
5. **DISTANCE_MAXIMAL_DOOR** : Distance between the robot and the entry. Used to avoid reducing the power of a robot's reboot.
6. **MOVE_MINIMUM** : Minimal distance for a move. It can tweak (increase/decrease) robot movement around the map.
7. **STOCHASTIC_1** : Probability of triggering reboots during aggregate phases.
8. **STOCHASTIC_2** : Probability for a robot to pick the vote or random color strategy in the central room.
9. **STOCHASTIC_3** : Probability of a robot to pick a random color when entering a room and the value if less than the best current value.
10. **DECISION_COLLECTIVE** : Quorum to reach for a vote to pass.

Now we have cite all parameters which play a different role in the program. As we can see, the length of this list could be huge. It could be interesting to found the most important criteria that we need to use during this selection which could be decide as best as possible all of this parameters.

3.2.2 Selection Criteria

We decided to choose the next criteria :

1. **Impact on the final result** : eliminate all parameters that could not be so important like MOVE_MINIMUM.
2. **Range of values to explore in the PSO algorithm** : eliminate the parameters with a range of exploration too big like all DISTANCE parameters. They not could be explored efficiently with PSO in 10 executions.

3. **Experimentally determined value** : eliminate all parameters that we found by trial and error like STOCHASTIC_1 which are need to have a value a little bit upper or equal to his actual value : 0.965 (otherwise a too much exploration could take place and make the PSO execution prohibitively long).

Along these criteria, we then keep only the following three parameters :

1. STOCHASTIC_2
2. STOCHASTIC_3
3. DECISION_COLLECTIVE

3.3 Determination of the number of particles

Now we have decided all parameters to improve we need to fund the initial parameters which could be improve our exploration to found the best parameters values.

When we observe the boxplot, we can see several good candidate values (15 particles, 20 particles, 25 particles, 30 particles and 35 particle) with a similar range of value and the best quality or lower value of steps. To choose between these three values, we decided to apply a Wilcoxon signed rank test.

Wilcoxon signed rank test The Wilcoxon test creates a pooled ranking of all observed differences between the dependent measurements. It uses the standard normal distributed z-value to test for significance(w3, 1999).

H_0 both samples are from the same population.

H_1 both samples are from a different population.

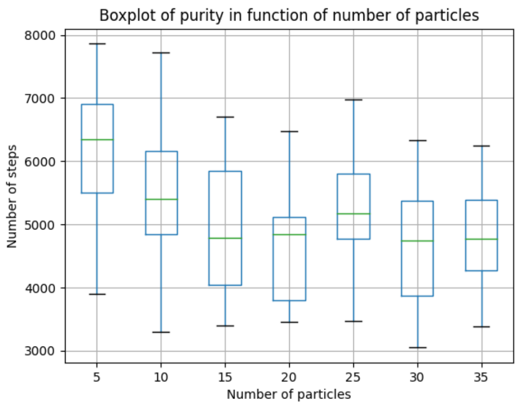

Figure 7: Execute PSO 20 times with 10 iterations for each number of particles

RanksumResult - Determination of particles numbers

Value	Value/5	Value/10	Value/15	Value/20	Value/25	Value/30	Value/35
5		1.8394	3.30	3.70586	2.7862	3.86816	3.705862
10	-1.8394		1.7312	2.0558	0.7303524	1.218	2.191
15	-3.30	-1.73		0.25697584	-1.21725	0.59510	0.3787
20	-3.70	-2.0558	-0.25697584		-1.704	0	-0.013
25	-2.786159	-0.73	1.217	1.7041556		1.7312	1.7041556
30	-3.868	-2.218	-0.5951	0	-1.7312		-0.10820
35	-3.70586	-2.191	-0.3787	0.0135	-1.704155	0.108	

The table "RanksumResult" help us to confirm the previous hypothesis than 20, 30 and 35 are the lowest value because he have the most higher value when we apply a comparison with 5 which are the most upper distribution. So we stopped to take in count the group of 15 and 25 particles because them value are lower than this three cohort) It could be noticed that 30 are the best of the set and values for 35 are quite close to 20 even if 20 have a value equal 0 with 30 which is underline an absence of any difference between this two cohort. Now, it could be interesting to look up the p-value to determine if this absence of distinction are true or not.

P value table

Value	Value/5	Value/10	Value/15	Value/20	Value/25	Value/30	Value/35
5		0.0658	0	0	0.005	0	0
10	0.0658554		0.0834	0.0398	0.46517	0.02655	0.02845
15	0	0.08		0.797	0.2235	0.5517775	0.7049
20	0	0.0398	0.797		0.08835	1.0	0.989
25	0.005	0.46517	0.2235	0.08835		0.0834	0.08835
30	0	0.02655	0.55177	1	0.083415		0.91383676
35	0	0.028	0.7049	0.9892	0.08835	0.913837	

With the table of p-values, we can observe than a population of 20 or 30 particles have a p-value upper than 0.05 (which are the threshold to reject or not hypothesis zero). So, we didn't reject the null hypothesis(Guillaume Broc, may 2018) and we can assume that the fact of the population of 20,25,30,35 could be are the same. Regarding the previous interpretation, we didn't take in count 25 and reject 35. By the fact that a 30-population is ideal and like we see that 20 particles and 30 particles have a same population, we didn't need to keep 35 particle population. It could be more interesting from an execution time point of view to pursue the project with a population size equals to 20 if we didn't loose in quality of results.

3.4 Determination of the value of both Phi parameters

When we observe the boxplot figure, we can notice that all solutions cover the same range of values. So to try and see any difference between them we need to apply a Wilcoxon signed rank test.

Wilcoxon signed rank test The Wilcoxon test creates a pooled ranking of all observed differences between the dependent measurements. It uses the standard normal distributed z-value to test for significance(w3, 1999).

H_0 both samples are from the same population.

H_1 both samples are from a different population.

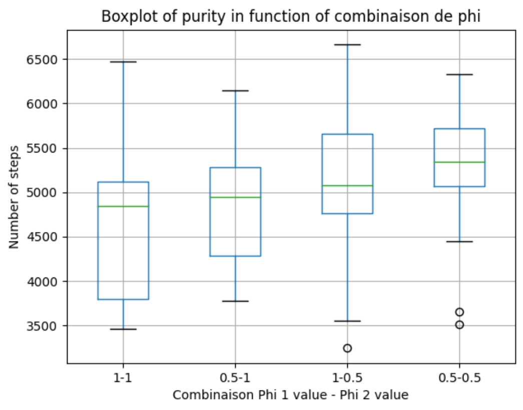

Figure 8: Execute PSO 20 times with 10 iterations for values of phi 1 and 2

RanksumResult - Determination of Phi's values

Value	Value/[1-1]	Value/[0.5-1]	Value/[1-0.5]	Value/[0.5-0.5]
1-1		-0.45985	-1.244304	-1.961
0.5-1	0.45985		-1.2984	-2.164
1-0.5	1.244	1.298		-0.595
0.5-0.5	1.9611	2.164	0.595	

The "RanksumResult" table helps us identify 1-1 and
0.5-1 as having the best higher scores in comparison of the other cohort. We decide to keep only this two cohort and look up the table to see if any difference exist in between them.

P value

Value	Value/[1-1]	Value/[0.5-1]	Value/[1-0.5]	Value/[0.5-0.5]
1-1		0.6456	0.21338766	0.04986
0.5-1	0.645622		0.1941	0.03
1-0.5	0.21339	0.194148		0.551775
0.5-0.5	0.04986	0.0304638	0.55177	

We observe a non-reject of the null hypothesis for the comparison of 1-1 and 0.5-1 by a p-value upper than 0.05(Guillaume Broc, may 2018). It could be from the same population, so if we choose one or the other, both could be produce the same result. We decided to keep only the combination 0.5-1 which are obtain the best value in the RanksumResult even if by the actual result, we know that could be make any difference in the production of the next result if we choose 1-1 or 0.5-1 because we know they have the same population, same range of value.

4 Comparison of both techniques

In the previous part, we underlined the different parameters which performed best: a number of particles equals to 20, a value of phi 1 equals to 0.5 and a value of phi 2 equals to 1. We looking in the all results are be produce with parameter and we decided to take the combination of parameters (STOCHATIC_2, STOCHASTIC_3 and DECISION_COLLECTIVE) which are produce the best result. The best results product by this parameters is 3773 steps with the next combination of parameters :

1. **STOCHASTIC_2** : 0
2. **STOCHASTIC_3** : 0.86
3. **DECISION_COLLECTIVE** : 1

To analyze the performance gain obtained by optimizing these parameters, we can run PSO with the optimized and non-optimized parameters, and compatre the results.

4.1 Results

We can observe a superposition of both ranges but also that PSO with non-optimized parameters has a lower mean. To confirm the fact there is a superposition we need to apply a Wilcoxon signed rank test.

Wilcoxon signed rank test The Wilcoxon test creates a pooled ranking of all observed differences between the two dependent measurements. It uses the standard normal distributed z-value to test of significance(w3, 1999).

H_0 both samples are from the same population.

H_1 both samples are from a different population.

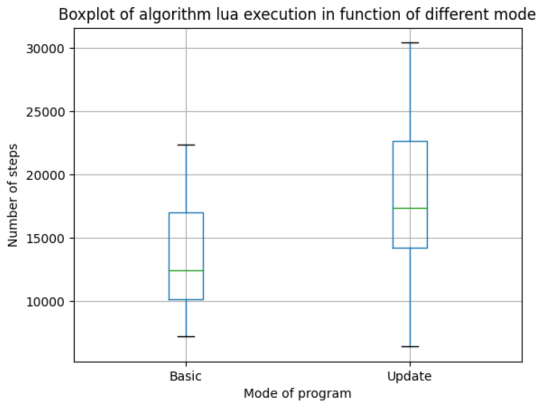

Figure 9: Execute PSO 20 times with 10 iterations for optimized and non-optimized parameters

Mode	P-value	RanksumResult
Basic in function Update	0.007	-2.6779588

We reject the null hypothesis zeros because the p-value are lower than 0.05(Guillaume Broc, may 2018). So we can claim that the initial program has a lower population than the updated program. So in conclusion, PSO doesn't improve my program.

It could be due by the fact that 10 iterations are not sufficient to find the best combinations of values for 3 parameters, or it could perhaps be more interesting to take into account much more parameters and explore more correctly the space for dimensions of parameters. During this execution, we always used "fully connected" topology, so It could be interesting to try repeat 10 iterations (or more) of PSO with different typology's.

References

Carlo Pinciroli, R. O. G. P. A. B. M. B. N. M. E. F. G. D. C. F. D. M. B. L. M. G. M. D., Vito Trianni. (2012). *The argos website.* Retrieved from `https://www.argos-sim.info/index.php`

de Almeida, B. S. G., & Leite, V. C. (May 30th 2019). *Particle swarm optimization: A powerful technique for solving engineering problems.* Retrieved from `https://www.intechopen.com/books/swarm-intelligence-recent-advances-new-perspectives-and-applications/particle-swarm-optimization-a-powerful-technique-for-solving-engineering-problems`

Guillaume Broc, B. C. (may 2018). *Analyse de données.* Retrieved from `https://books.google.be/books?id=6sxaDwAAQBAJ&pg=PA161&lpg=PA161&dq=wilcoxon+itnerpr%C3%A9tation+rejet+h0&source=bl&ots=rCoNZl60lj&sig=ACfU3U0EuFsiCRzKe-23nF_uzwAdeSah8Q&hl=en&sa=X&ved=2ahUKEwi5ofrmku7pAhUNsKQKHQQ8BhgQ6AEwAHoECAoQAQ#v=onepage&q=wilcoxon%20itnerpr%C3%A9tation%20rejet%20h0&f=false`

w3. (1999). *Conduct and interpret a wilcoxon sign test.* Retrieved from `https://www.statisticssolutions.com/wilcoxon-sign-test/`