The Spatial Distribution of Agents in Virtual Terrains through Search and Optimization Algorithms

Felipe G. Pires
Undergraduate Program in Computer Engineering
Federal University of Santa Maria – UFSM
Santa Maria – RS, Brazil
felgpires@gmail.com

Marcos S. Morgenstern
Graduate Program in Computer Science
Federal University of Santa Maria – UFSM
Santa Maria – RS, Brazil
marcossmorgenstern@gmail.com

Edison P. Freitas
Graduate Program in Computer Science
Federal University of Rio Grande do Sul – UFRGS
Porto Alegre – RS, Brazil
edisonpf@gmail.com

Luis A. L. Silva
Graduate Program in Computer Science
Federal University of Santa Maria – UFSM
Santa Maria – RS, Brazil
luisalvaro@inf.ufsm.br

***Abstract*– The use of meta-heuristics in the resolution of multi-agent spatial reasoning problems is not a mature Artificial Intelligence and Games research field. To approach the problem of positioning agents in virtual terrains according to a required formation pattern, this paper details how to model this problem in a state-space search and optimization framework. With particular attention to strict processing time and domain constraints, then it analyzes the effectiveness of optimization algorithms with different nature: the Simulated Annealing and the Tabu Search algorithms. Experimental results indicate that these methods are able to achieve high optimization rates and to provide low error spatial agent distribution solutions.**

***Keywords*– Spatial agent distribution, Meta-heuristics; Games**

I. INTRODUCTION

Meta-heuristics include heuristic search methods used to solve optimization problems. Based on limited processing time and domain constraints, such optimization methods have been studied in Real-Time Strategy games [1]. [2] uses a Declarative Programming approach to optimize the construction and location of buildings in a virtual battle scenario. [3] explores a Genetic Algorithm that uses a battle specification as input and optimizes the location of buildings in a virtual battle scenario. [4] uses a Best-First Search algorithm to investigate a search-space filled with Potential Fields as an alternative to solve the problem of spatial arrangement of buildings. [5] explores a local search method where the Adaptive Search algorithm is used to solve the problem of optimizing the construction of protecting barriers (walls). Despite such efforts, the use of search and optimization algorithms in multi-agent game environments is not a mature Artificial Intelligence and Games research field. Further work is still required in different fronts.

This paper investigates the spatial distribution of terrestrial agents in the virtual terrain environment as a problem of spatial reasoning [6] and combinatorial optimization. To do so, the work explores meta-heuristic algorithms of different nature: the *Simulated Annealing (SA)* and the *Tabu Search (TS)* algorithms [7, 8]. While the SA is a single-solution iterative stochastic algorithm that starts with some arbitrary solution and does not use information collected during the search, the TS is a single-solution meta-heuristic including various types of memory structures with the purpose of escaping from local optimum locations. This work analyzes the effectiveness of these algorithms in the resolution of a spatial agent distribution problem, where agents need to be positioned in selected areas of the virtual terrain according to a required formation pattern. In addition to considering domain constraints, the algorithms were subject to 1s and 10s strict processing time limits, where the tests evaluated the error and the optimization rate of the generated spatial agent distributions.

II. A MODEL FOR THE AGENT DISTRIBUTION PROBLEM

A user interacting with the virtual terrain environment is responsible for making the initial reconnaissance of the terrain region in which agents have to be deployed. Within this user-selected deployment area, the individual position of each agent is determined by the presence of static terrain features of interest to which the agents are positioned close to. This is a strong constraint in this spatial reasoning problem. It is illustrated in Fig. 1. We intentionally used the "terrain feature" term in this paper. That is because these features can largely vary from one application problem to another. In effect, such features can be instantiated as buildings, mountains, roads, courses of water, vegetation types (used in our work), and so on. From an arbitrary initial agent distribution in this area, combinations of individual agent positions are evaluated during the search and optimization process. The goal is to search for the optimal positions for the agents as far as a targeted formation pattern is concerned. This spatial reasoning problem is modeled as:

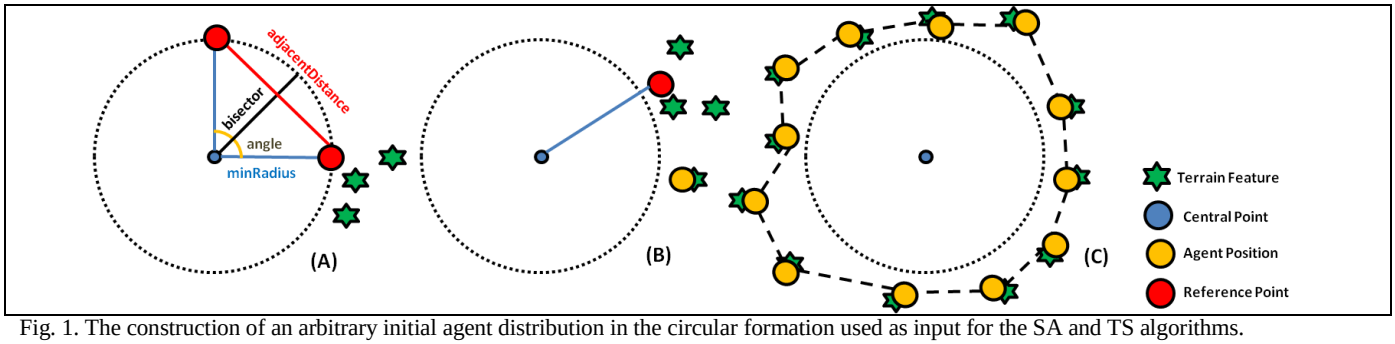

Fig. 1. The construction of an arbitrary initial agent distribution in the circular formation used as input for the SA and TS algorithms.

i) **State structure**: For *n* agents (Ag_0, Ag_1 , ..., Ag_n) in the virtual environment, the *List<Vector2>* named posAg[n] is used. This structure records the position of each agent in the virtual terrain. ii) **State transition operator**: A neighborhood function N is a mapping N: $S \rightarrow 2^S$ that assigns each solution s $\in$ S to a set of solutions N(s) $\subset$ S. A solution s' in the vicinity of s (s' $\in$ N(s)) is called neighbor of s. A neighbor is generated by the execution of a moveAg() operator that performs a small disturbance in the current solution s. The moveAg() alters the current Ag_i position to another one, where positions are determined by the terrain features in the vicinity of the current agent position (Fig. 1). iii) **Solution quality**: The objective function f(x) formulates the objective to be achieved. It associates each solution in the search space with a real value describing the quality or suitability of the solution, f: S $\rightarrow$ R. Then, it allows a complete ordering of all solutions in the search space. The objective function guides the search toward "good" solutions.

In this work, agents are spatially distributed into the selected terrain area according to a circular formation pattern. Thus, the objective function measures the circularity of the agent distribution solution. The circularity is such that $C = 4\pi A/P^2$, where C is the circularity, A is the area and P is the perimeter. The individual position of each agent is linked to the presence and consequent position of terrain features in the area in which the agents have to be located. For 12 agents, for example, it is possible to calculate the minimum radius of the circular formation required to ensure that agents are, for example, at least 10m from each other, as stated by the underlying domain constraint (Fig. 1). A perfect arbitrary circle with two points randomly positioned on the circle outline results in a triangle (Fig. 1 (A)). In it, there are two edges with the same magnitude which is the minimum formation radius (minRadius); the edge is defined by the minimum distance (10m) between adjacent agents. To place more points at the circle outline, it is used a fixed angle for each additional point. This angle (e.g. 360º/(12 agents) = 30º) ensures that each agent will be at a valid distance between each other. The tracing of a bisector at the center of the arbitrary circle results in a triangle rectangle (Fig. 1 (A)), allowing the use of trigonometric functions. The minimum radius value to ensure the required distance between the adjacent agents in this spatial distribution is such that *sine*((angle = 30º) / 2) = ((adjacentDistance = 10m) / 2) / minRadius, resulting on the minRadius of 19.31m.

The minimum circular formation radius and the positioning angle of the agents are not sufficient to build a valid initial spatial distribution for the agents. To do this, a center point is selected and used as a reference point in which the distance is defined by the minimum radius. From this point, the closest terrain feature that satisfies the domain restrictions is selected (Fig. 1 (B). With the terrain feature, an agent is positioned at that location. From the position of the first agent, the algorithm updates the angle of the first point in order to find the next reference point. From there, the algorithm searches for another terrain feature that is present at the vicinity of that location, permitting to determine a position to the second agent. The process continues until an initial agent distribution is found (Fig. 1 (C)). If there are no terrain features of interest in the searched locations, the agent positioning process restarts elsewhere in the selected virtual terrain area.

III. The Search and Optimization Process

The SA is a heuristic for global optimization [7, 8]. From an initial agent distribution solution, captured as a state in this state-space search and optimization problem, a new solution in the vicinity of the current solution is generated according to the algorithm iteration. If the new solution is better, it replaces the current solution. If it is worse, it will be accepted according to a certain probability that depends on the current temperature value. With the algorithm execution, the probability of accepting such kind of state transition move (opposing the optimization direction) decreases until the algorithm allows substitutions only in direction to better agent distribution solution states.

In the SA (Algorithm 1), the cooling rate and temperature parameters are adjusted. The appropriate initial temperature (t_0) is specific to the problem, and is estimated from an analysis in which all solution state quality increments are accepted permitting to calculate the average increase. Conditional variables are checked during the optimization process. If the temperature variable reaches zero, the optimization is finished. The time constraint variable guarantees that the state does not get stuck in a solution that does not have a quality improvement. From an initial solution and a list of terrain features in the regions where agents have to be distributed, a group of neighboring solution states is generated. Neighboring solutions are formed by modifying the position of a randomly chosen agent, which is positioned next to another terrain feature (Fig. 1). From this, all generated neighboring solutions are

evaluated individually. When one of these neighboring solutions has a better quality than the initial solution, the neighboring solution is attributed to the current solution. When a selected neighbor solution does not improve on the initial solution, the probability of acceptance is verified through the *Boltzmann* distribution.

Algorithm 1. The *Simulated Annealing* algorithm.

```
1:  algorithm Simulated Annealing()
    input:initialState(s),timeConstraint, terrainFeature(searchList)
    output:bestState(s_Best)
2:  s_Best ← s
3:  Cooling ← 0.95
4:  Temperature ← SetTemperature()
5:  while temperature > 0 and timeConstraint > 0 do
6:      N[s]← GenerateNeighborhood(s_i,searchList)
7:      forall s in N[s] do
8:          s' ← s
9:          if f(s') > f(s_Best) then
10:             s_Best ← s'
11:         else
12:             Δ ← f(s') - f(s_Best)
13:             if EΔ/k*temperature > Random(0,1) then
14:                 s_Best ← s'
15:             end if
16:         end if
17:     end for
18:     temperature ← temperature * Cooling
19: end while
20: end algorithm
```

The probability of accepting a worse neighboring solution is proportional to the temperature T. At high temperatures, the probability of accepting worse state transition movements is high. If $T = \infty$, all movements are accepted. At low temperatures, the probability of accepting a worse movement decreases. If T = 0, no worse movement is accepted. Therefore, the probability of accepting a large deterioration in the solution quality decreases exponentially towards 0. When all neighboring solutions are evaluated, the temperature is decreased. The SA continues until the termination conditions are met.

The TS algorithm [7, 8] seeks to improve the search by accepting state transition moves that do not improve the quality of the solution. The search space contains all the possible solutions considered during the search. The neighborhood structure is a set of solutions in this search space. The method explores the solution space by moving the current solution to another one. To do so, it uses a tabu-list to record the movement of a solution examined recently. If the algorithm accepts the solution that does not have the best quality, the tabu-list mechanism prevents the formation of search cycles.

The initial configuration of the TS method (Algorithm 2) assigns the initial agent distribution solution to a variable and qualifies it to record the best found solution. Because this initial state is visited early in the algorithm, the tabu-list is initialized with it. The tabu-list is a short-term memory structure that records the visited states. The algorithm searches for a solution until a specified stop condition is achieved. Neighboring solutions are generated with the random repositioning of a chosen agent to terrain feature positions near to an ideal agent position (Fig. 1). Once the neighborhood structure is generated, the first element of the neighborhood is assigned as having the best quality.

Algorithm 2. The *Tabu Search* algorithm.

```
1:  algorithm Tabu Search()
    input:initialState(s),timeConstraint, terrainFeature(searchList)
    output:bestState(s_Best)
2:  s_Best ← s
3:  tabuList.push(s_Best)
4:  while timeConstraint > 0 do
5:      N[s]← GenerateNeighborhood(s_Best,searchList)
6:      s_bestCandidate ← N.firstElement
7:      forall s_c in N[s] do
8:          if ((( not tabuList.contains(s_c) and
                                  (F(s_c) > F((s_bestCandidate)))) then
9:              s_bestCandidate ← s_c
10:         end if
11:     end for
12:     if f(s_bestCandidate) > f(s_Best) then
13:         s_Best ← s_bestCandidate
14:     end if
15:     tabuList.push(s_bestCandidate)
16:     if (tabuList.Size > maxTabuSize) then
17:         tabuList.RemoveFirst()
18:     end if
19: end while
20: end algorithm
```

The tabu-list is implemented so that the search is not overly restrictive in its ability of searching for better solutions. In certain situations, the condition of belonging to the tabu-list can prohibit state-space movements toward a global optimum. Therefore, conditions to cancel the tabu, called as aspiration criterion, are necessary. This criterion allows a change of the current solution, even if it belongs to the tabu-list. For each neighborhood solution, if these states do not belong to the tabu-list and they have the best quality among the other neighborhood solutions, they are called the best solution in that neighborhood. According to the aspiration criterion, if the neighborhood search has found a better solution, it is taken as the best solution. The best local candidate is always added to the tabu-list. If the tabu-list is full, some of its elements may expire. Thus, the procedure selects the best local candidate even if it has worse quality than the best found agent distribution solution.

IV. Experiments and Results

The experiments analyze the effectiveness of the SA and TS algorithms in the optimization of the agents' distributions in the virtual terrain. The tests were carried out in 500 different virtual terrain areas. There, each algorithm executed the search and optimization of solutions referring to different numbers of agents.

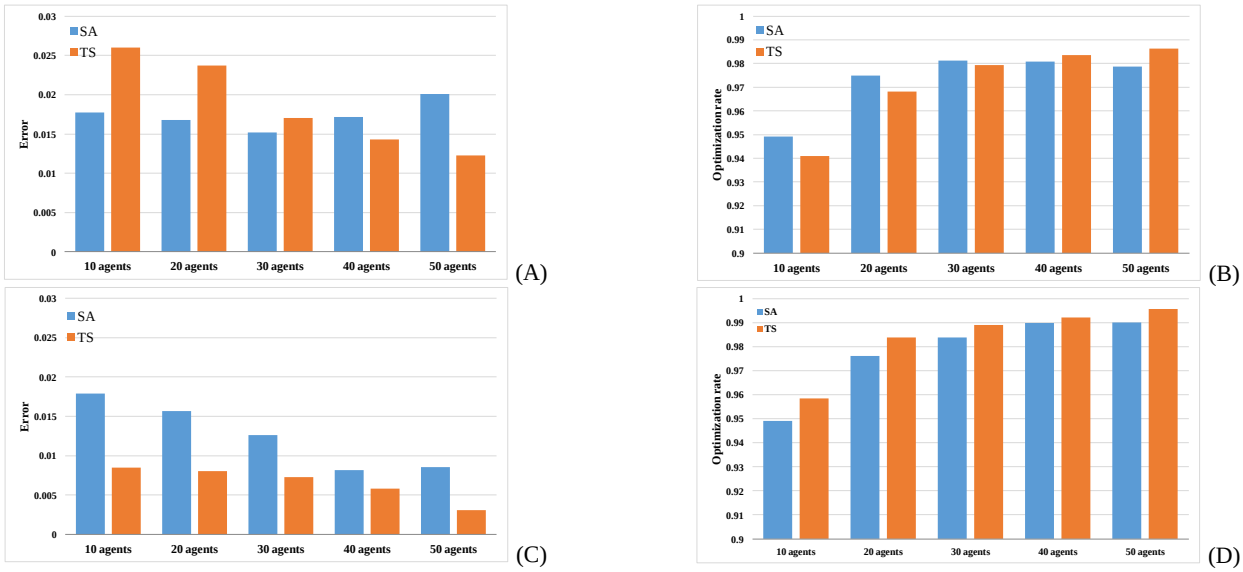

Fig. 2. The average error and optimization rate within 1s and 10s for the resulting spatial agent distribution solutions.

Within the 1s and 10s (according to [9], users will increasingly feel more disconnected from the system execution if they don't get a response in less than 10s), starting from valid initial agent distributions (the ones in which the domain constraints are satisfied – Fig. 1 (C)), the algorithms modifies the agent positions to obtain the best possible spatial distribution. The evaluation metrics used in the tests are: i) the average agent distribution error and (ii) the average optimization rate for the algorithms (i.e. mean values computed from the best results obtained in each one of the 500 terrain areas). The tests were executed on the Unity Engine version® 2019.2.2f1, running on the Intel Core i7 6700 HQ CPU @2.60Ghz processor, 16GB, and Windows 10. Fig. 2 presents test results. Within 1s, the TS error decreased with the increase of the number of agents. It was not possible to observe similar behavior (decrease / increase) on the SA error. Up to 30 agents, the TS error was higher than the SA one. For 40-50 agents, this TS error was lower than the SA one. Within 10s, the SA error decreased up to 40 agents, where it remained stable with 40-50 agents. The TS error had a small decrease in relation to the increase of the number of agents. Within 1s, the SA optimization rate for 10 agents was slightly reduced, while it increased and then remained constant with 20-50 agents. Similar reduction was observed with the execution of the TS with 10-20 agents, and a constant higher optimization rate for the TS was observed with 20-50 agents. Up to 30 agents, the SA optimization rate was higher than the TS rate, while it was lower than the TS with 40-50 agents. Within 10s, the optimization rate for both algorithms increased with the increase of the number of agents. In conclusion, both algorithms presented satisfactory agent distribution results within 1s and 10s. Both SA and TS errors diminished with the increase of the processing time limit. The TS presented a high error decrease from 1s to 10s. For all number of agents, the TS optimization rate was higher than the SA optimization rate. From 1s to 10s, there was an improvement on the TS optimization rate, while similar SA improvement was only observed with 40-50 agents.

V. Final Remarks

This paper investigates the effectiveness of meta-heuristics in the resolution of a multi-agent spatial distribution problem. It details how to model this state-space search and optimization problem in order to obtain solutions with the SA and TS algorithms. With strict processing time limits, the tested algorithms achieved spatial distributions for the agents in different virtual terrain scenarios, where the solutions were close to the ideal formation pattern. All in all, this work so far presents promising results in our project, where a deeper experimental analysis of this research problem is still under-investigation. In addition, future work will approach similar spatial reasoning problems with: the combined use of static and dynamic terrain features, the analysis of other kinds of formation patterns and the exploration of hybrid meta-heuristic approaches.

References

[1] S. Ontanón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss, "A survey of real-time strategy game ai research and competition in starcraft," *IEEE Trans. on Computational Intelligence and AI in games*, vol. 5, pp. 293-311, 2013.

[2] M. Certicky, "Implementing a Wall-In Building Placement in StarCraft with Declarative Programming," arXiv:1306.44602013.

[3] N. A. Barriga, M. Stanescu, and M. Buro, "Building placement optimization in real-time strategy games," in *Conf. on AI and Interactive Digital Entertainment (AIIDE 2014)*, Raleigh, NC, USA, 2014, pp. 2-7.

[4] C. F. Oliveira and C. A. G. Madeira, "Creating efficient walls using potential fields in real-time strategy games," in *Comp. Intelligence and Games (CIG 2015)*, Tainan, Taiwan, 2015, pp. 138-145.

[5] F. Richoux, A. Uriarte, and S. Ontanón, "Walling in Strategy Games via Constraint Optimization," in *Conf. on AI and Interactive Digital Entertainment (AIIDE 2014)*, Raleigh, NC, USA, 2014, pp. 52-58.

[6] K. D. Forbus, J. V. Mahoney, and K. Dill, "How qualitative spatial reasoning can improve strategy game AIs," *IEEE Intelligent Systems*, vol. 17, 2002.

[7] I. BoussaïD, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information sciences*, vol. 237, pp. 82-117, 2013.

[8] E.-G. Talbi, *Metaheuristics: from design to implementation* vol. 74: John Wiley & Sons, 2009.

[9] J. Nielsen, *Usability engineering*: Elsevier, 1994.