# Multidirectional Physarum Solver: an Innovative Bio-inspired Algorithm for Optimal Discrete Decision Making

Luca Masi[1,*]

*University of Strathclyde, 75 Montrose Street G1 1XJ, Glasgow, UK*
*Tel.: 0141 548 2083*
*Fax: 0141 552 5105*

Massimiliano Vasile[2]

*University of Strathclyde, 75 Montrose Street G1 1XJ, Glasgow, UK*
*Tel.: 0141 548 2083*
*Fax: 0141 552 5105*

**Abstract**

This paper introduces a new bio-inspired algorithm for optimal discrete decision making, able to incrementally grow and explore decision graphs in multiple directions. The heuristic draws inspiration from the idea that building decision sequences from multiple directions and then combining the sequences is an optimal choice if compared with a unidirectional approach. The behaviour of the slime mould *Physarum Polycephalum*, a large single-celled amoeboid organism, is used as basic heuristic for graph exploration and growth. The algorithm is here applied to the solution of classical problems in operational research, i.e. symmetric Travelling Salesman and Vehicle Routing Problems, with a number of cities ranging from 10 to 199. Simulations on selected test cases demonstrate that a multidirectional solver performs better than a unidirectional one. The ability to evaluate decisions from multiple directions enhances the performance of the solver in the construction and selection of optimal decision sequences.

*Keywords:*
Combinatorial optimization, Decision analysis, Physarum, Multidirectional, Traveling salesman problem, Vehicle routing problem

## 1. Introduction

Over the past two decades, bio-inspired computation has demonstrated to be an appealing approach to solve NP-hard problems in combinatorial optimization. As the complexity of the

*Corresponding author
*Email addresses:* luca.masi@strath.ac.uk (Luca Masi), massimiliano.vasile@strath.ac.uk (Massimiliano Vasile)
*URL:* http://www.strath.ac.uk/space/staff/lucamasi/ (Luca Masi), http://cms.mecheng.strath.ac.uk/t4/cmsstaffprofile.asp?id=214 (Massimiliano Vasile)
[1]PhD Candidate in Mechanical & Aerospace Engineering
[2]Reader in Space System Engineering and Associate Director at the Advanced Concept Laboratory

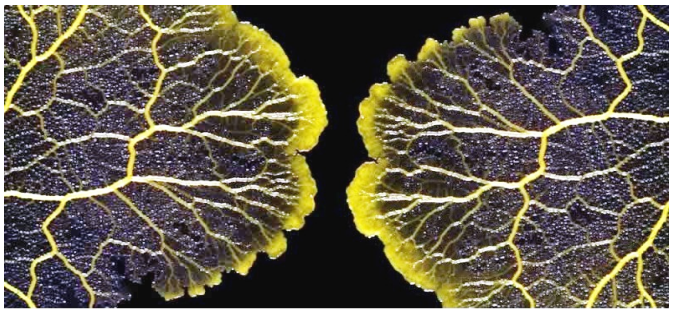


Figure 1: Two *Physarum Polycephalum* expanding in opposite directions. Image courtesy www.slickscience.com.

world is increasing due to exponentially growth of easily accessible information, the problem of finding the best solution over a maze of alternative options becomes intractable using classical algorithms in combinatorial optimization: biology inspired algorithms offer simple heuristics that have been proven to provide solutions to high dimensional decision making problems in a reasonable computational time [12]. Ant Colony Systems, Genetic Algorithms and Swarms are examples [14, 15, 8, 22, 20].

This paper introduces a new bio-ispired method for single objective discrete optimization. Discrete decision making problems are modeled with decision graphs where nodes represent the possible decisions while arcs represent the cost associated to each decision. Decision graphs are incrementally grown and explored in multiple directions, with the possibility of matching decision sequences. This concept will be further explained in Sect. 2.

This work aims at proving that a multidirectional incremental solver is more efficient, in terms of success rate at a fixed number of calls to the objective function (see Sect. 3.2), than a uni-directional incremental solver when applied to the solution of decision problems that can be represented with graphs where the contribution of an arc to a complete path can be evaluated moving forward or backward along the graph, here called reversible decision-making problems. This thesis will be experimentally demonstrated in Sect. 3.3 by solving a series of test cases. Symmetric Traveling Salesman (TSP) and Vehicle Routing Problems (VRP) with a number of cities between 10 and 199, introduced in Sect. 3, were chosen as representative examples of the above type of decision making problems.

The heuristic used for graph exploration and growth takes inspiration from a simple amoeboid organism, the *Physarum Polycephalum*, that is endowed by nature with heuristics that can be used to solve discrete decision making problems. It has been shown that *Physarum Polycephalum* is able to find the shortest path through a maze by changing its shape, connecting two food sources placed in the maze's entrance and exit [17]. In [25] it was shown that a living *Physarum* is able to recreate the Japan rail network in an experimental arena with food sources at each of the major cities in the region, and [2] has demonstrated that the *Physarum* is able to reproduce the designed highway network among several Mexican cities, with a slight mismatch. *Physarum* based algorithms have been developed recently to solve multi-source problems with a simple geometry [13, 24], mazes [23] and transport network problems [25, 23].

The algorithm proposed in this paper has been used previously in [16] to solve small-scale (10 to 16 cities) TSP and VRP problems. These problems will be further discussed in Sect. 3 along

$$\begin{cases} Q_{12} \sim \dfrac{r_{12}^4}{L_{12}} \Delta p_{12} \\ P_{12} = \dfrac{Q_{12}}{Q_{12} + Q_{13}} \end{cases}$$

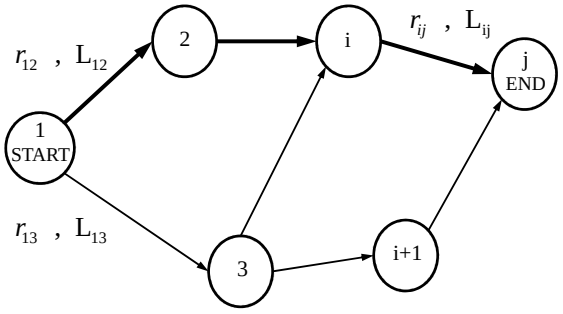



Figure 2: Figure showing a simple graph: thicker arrows represent higher fluxes. In this example $Q_{12} > Q_{13} \Rightarrow P_{12} > P_{13}$.

with the application to larger scale TSP and VRP problems.

## 2. Multidirectional incremental modified *Physarum* algorithm

Drawing inspiration from the behaviour of *Physarum Polycephalum*, a mathematical model for multidirectional discrete decision making is proposed in this section. Millions of years of evolution have provided this humble organism with appealing heuristics that can help humans to solve hard decision making problems.

### *2.1. Physarum biology*

*Physarum Polycephalum* is a large, single-celled amoeboid organism that exhibits intelligent plant-like and animal-like characteristics. Its main vegetative state, the *plasmodium*, is formed of a network of veins (*pseudopodia*). The stream in these tubes is both a carrier of chemical and physical signals, and a supply network of nutrients throughout the organism [24]. *Physarum* searches for food by extending this net of veins, whose flux is incremented or decremented depending on the food position with reference to its centre. The longest is the path connecting the centre with the source of food, the smallest is the flux.

### *2.2. Reversible decision making problems*

Directed graphs can be used to model reversible discrete decision problems, i.e. problems where for each decision that induces a change from state *a* to state *b*, whose cost is *C(a,b)*, it is possible to evaluate an inverted decision that brings back from *b* to *a*, whose cost is *C(b,a)*. Here the interest is in general graphs in which the cost *C(a,b)* may not be identical to the cost *C(b,a)* but the decision that brings from *b* to *a* exists, can be evaluated and corresponds to the same solution that brings from *a* to *b*. These types of graph can be seen as the superposition of two directed graphs (direct-flow, *DF*, and back-flow, *BF*, graphs) whose nodes are coincident and edges have opposite orientation. In so doing, the decision between state *a* and *b* has a forward

link $a$ to $b$ and a superposed backward link $b$ to $a$. It is assumed that the first decision node is the heart of a growing *Physarum* in *DF*, and the end decision node the heart of a growing *Physarum* in *BF*. The two *Physarum* are able to incrementally grow the decision graph in the two directions by extending their net of veins. A multiple direction growing decision *Physarum* graph is finally obtained. In the example mentioned before, the *Physarum* working in DF would build its graph by creating arcs that move from $a$ to $b$. If the graph was traversed by a virtual agent, the agent would walk along an arc from $a$ to $b$. The other *Physarum* would build its graph in the opposite direction then walking along each arc from $b$ to $a$. The result is a graph where both nodes and links are incrementally built by two expanding *Physarum*.

If one introduces the further assumption that the cost associated with the decision between state $a$ and $b$ does not have dependencies on all the previous decisions, then choosing a mesh network topology for the graph is optimal: in fact by considering each possible state as a unique node of a growing network, replication of nodes is avoided (this would not be avoided using a tree topology), nodes' number is minimized as well as the maximum number of links. Note that this assumption applies to the case of a decision maker dealing with symmetric TSP and VRP but it is not a requirement for the method proposed in this paper.

It should be noted that, under the previous assumptions, for TSP and VRP problems the number of nodes is fixed (the number of cities, see Sect. 3) and it is not incremented by the algorithm. The *Physarum* algorithm builds only new links during the optimization process. On the other hand, the capacity of nodes' creation is useful when solving decisional problems where the solution is built by taking possible single decisions (nodes) from a large database or decision space and the complete decision sequence is a subset of the decision space. This feature will be demonstrated in future works.

Fig. 1 shows two living *Physarum* growing in opposite directions. It should be noted that considering the two possibilities that a link between two nodes exists or does not exist, if the graph is sufficiently large (nodes $\geq 6$), for the Ramsey's theorem a monochromatic (link or no link) complete subgraph exists and its order is 3 (a triangle). This consideration allows one to say that during the graph growth there is at least a 50% probability that a connected triangle exists, i.e. there is at least a 50% probability that a closed decisional sequence exists. Although during the graph growth closed loops are not directly built, they may be present: the mathematical nature of a growing mesh network topology leads to possible loops presence that a developer would like to avoid for certain decisional problems, i.e. Hamiltonian paths. A solution in order to avoid loops during graph exploration, as made for the *Physarum* algorithm, is the introduction of a memory associated with exploring agents, i.e. the introduction of a constraint for each movement of the agent on the graph, based on previous sequence of decisions, coupled with the possibility of building new arcs in case of loops.

### *2.3. Algorithmic*

The mathematical model of the algorithm is composed of two main parts: decision network exploration and growth in multiple directions. They are presented in this section along with an algorithm's restart procedure.

*Decision network exploration.* The flux through the net of *Physarum* veins can be modeled as a classical Hagen-Poiseuille flow in cylindrical ducts with diameter variable with time [25, 13, 24, 23]:

$$Q_{ij} = \frac{\pi r_{ij}^4}{8\mu} \frac{\Delta p_{ij}}{L_{ij}} \tag{1}$$

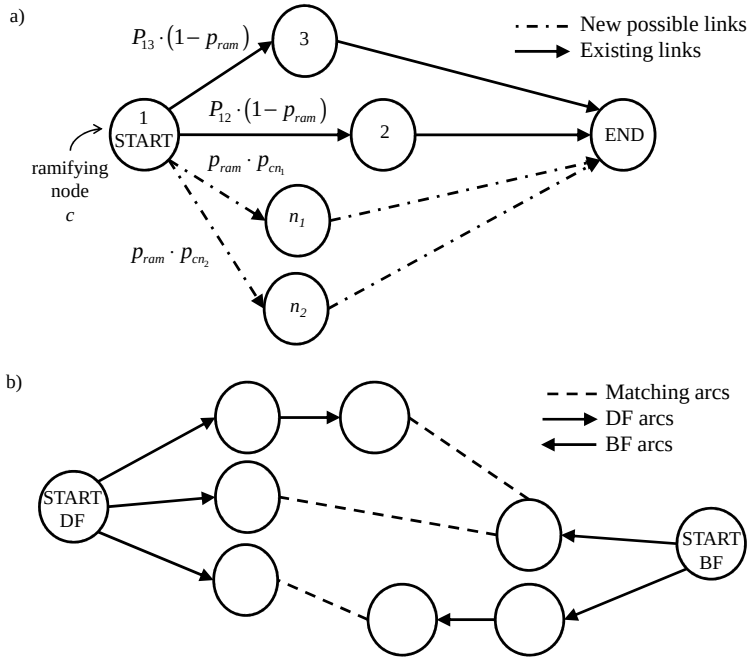



Figure 3: Figure showing ramification towards a new node (a), and matching between decision paths in DF and BF (b).

where $Q_{ij}$ is the flux between $i$ and $j$, $\mu$ is the dynamic viscosity, $r_{ij}$ the radius, $L_{ij}$ the length and $\Delta p_{ij}$ the pressure gradient. In Fig. 2 these quantities in a simple graph are shown. Diameter variations allow a change in the flux. Veins' dilation due to an increasing number of nutrients flowing can be modeled using a monotonic function of the flux:

$$\frac{d}{dt} r_{ij} \Big|_{dilation} = f\left(Q_{ij}\right) \tag{2}$$

where $f(0) = 0$ , i.e. linear, sigmoidal, etc. It can be assumed that the dynamics of tube adaptation are sufficiently slow for the flow to be considered at steady state [24]. Veins' contraction due to evaporative effect can be assumed to be linear with radius:

$$\frac{d}{dt} r_{ij} \Big|_{contraction} = -\rho r_{ij} \tag{3}$$

where $\rho \in [0, 1]$ is defined evaporation coefficient.

The probability associated with each vein connecting $i$ and $j$ is then computed using a simple adjacency probability matrix based on fluxes:

$$P_{ij} = \begin{cases} \frac{Q_{ij}}{\sum_{j \in N_i} Q_{ij}} & if\ j \in N_i \\ 0 & if\ j \notin N_i \end{cases} \tag{4}$$

Table 1: Setting parameters for the modified *Physarum* solver.

| | |
|---|---|
| $m$ | Linear dilation coefficient, see Eq. (9). |
| $\rho$ | Evaporation coefficient, see Eq. (3). |
| $GF$ | Growth factor, see Eq. (5) |
| $N_{agents}$ | Number of virtual agents. |
| $p_{ram}$ | Probability of ramification, see Sect. 2.3. |
| $\alpha$ | Weight on ramification, see Eq. (6). |

where $N_i$ is the set of neighbour for $i$.

A further social term in the dilation process was added in the algorithm and takes inspiration from the behaviour of the amoeba *Dictyostelium discoideum* (Dd). In its aggregative and slug stages, amoebae are chemotactically sensitive to cAMP signals (a chemical known as cyclic Adenosine Monophosphate [14]). A starving pacemaker amoeba starts to emit cAMP, that is a call for aggregation and subsequent collective behaviour. In a computational algorithm, pacemaker can be considered the agent with best objective function. A linear dilation for the pacemaker, which is defined as the best path so far in the decision graph in terms of objective function, was here chosen:

$$\frac{d}{dt} r_{ij_{best}} \bigg|_{elasticity} = GF r_{ij_{best}} \tag{5}$$

where $GF$ is the growth factor of the best chain of veins and $r_{ij_{best}}$ the veins' radii. This pacemaker call can be interpreted as a variable elasticity of the veins with time: best veins increase their capacity of dilation with a percentage $GF$. This is an additive term in the veins' dilation process, whose first main term is expressed in Eq. (2).

*Unidirectional decision network growth.* The incremental growth of the decision network is then based on a weighted roulette. Nutrients inside veins are interpreted as virtual agents that move in accord with the adjacency probability matrix in Eq. (4) on the existing graph. Once they are at a node, there is an *a priori* probability $p_{ram}$ of ramification towards new nodes that are not yet connected with the present node. If ramification is the choice, a weighted roulette, based on objective function evaluations, helps the *Physarum* with the selection and construction of a new link. The probability of a new link construction from the current node $c$ to a new possible node $n_i \in N$, where $N$ is the set of new possible decisions, is here assumed to be inversely proportional to the cost $L_{cn_i}$ of the decision between $c$ and $n_i$:

$$p_{cn_i} \propto \frac{1}{L^{\alpha}_{cn_i}} \tag{6}$$

where $\alpha$ is a weight. Once a new link is built, a complete decision path is constructed (creating other links if necessary). Fig. 3(a) shows a possible ramification in a simple graph; dotted lines represent feasible arcs not yet existing: if an agent is in at the start node, which represents the current possible ramifying node $c$, it has a probability $p_{ram}$ of ramification towards new nodes. If the agent decides to create a new link, a new node is selected according to Eq. (6). If the agent decides to explore the existing branches then the *decision network exploration* procedure is followed. This process will continue till the end node is reached. In the case shown in Fig. 3(a)

**Algorithm 1** Multidirectional incremental modified Physarum solver

initialize $m$, $\rho$, $GF$, $N_{agents}$, $p_{ram}$, $\alpha$
generate a random route from start to destination both in $DF$ and $BF$
**for** each generation **do**
  **for** each virtual agent in all directions ($DF$ and $BF$) **do**
    **if** current node $\neq$ end node **then**
      **if** $rand \leq p_{ram}$ **then**
        using Eq. (6) create a new decision path, building missing links and nodes
      **else**
        move on existing graph using Eq. (4).
      **end if**
    **end if**
  **end for**
  look for possible matchings, see Sect. 2.3.
  contract and dilate veins using Eqs. (2), (3), (5)
  **if** $r_{ij}$ exceeds upper radius limit, see Eq. (10) **then**
    block radius increment
  **end if**
  update fluxes and probabilities using Eqs. (1), (4)
  **if** *restart condition* **then**
    update veins' radii using Eq. (7)
    update fluxes and probabilities using Eqs. (1), (4)
  **end if**
**end for**

when an agent is at the start node, it can explore the already linked nodes 2 and 3 or generate a new link. Once at node 2 or 3 the only possibility in order to complete the decision path is a new link construction between the current node and ending node.

*Multidirectional decision network growth.* If multiple *Physarum* are simultaneously growth one can explore the decision space from multiple directions. Here a bi-directional approach is presented in which two *Physarum*, called *DF* and *BF*, form a network made of two superposed graphs. While growing the two expanding *Physarum* have the possibility of matching decision sequences: agents can build and traverse arcs that connect nodes belonging to *DF* and *BF* *Physarum* respectively forming a single path from the heart of one *Physarum* to the heart of the other *Physarum*. In Fig. 3(b) simple cases of matching in a graph are shown: the matched decision path is given by the union of a route in the *DF* and one in the *BF* through a matching arc. Several types of matching strategies are possible. The following types of matching strategies, named D&B with *x*M where the prefix *x* indicates the type of matching selected, were implemented and tested. *All-matching* (*a*M), in which all joint paths are saved and evaluated, *random-matching* (*r*M), in which some joint paths are selected among all the possible joint paths with a probabilistic rule and *selective-matching* (*s*M), in which some joint paths are selected among all the possible joint paths according to an elitist criterion: during each generation, the elitist criterion selects a joint path if and only if it is better in terms of total cost than the previous joint paths selected during the same generation. These strategies will be compared

in the *Tuscany10* test case, see Sect. 3.3. It could be noted that if a high number of exploring agents is chosen, a high number of paths are matched. This could lead to a slowdown of the code speed, especially if complete decision sequences are long, as in more complex VRP and TSP (more than 20 cities). For this reason, an other strategy, called *improved selective-matching* was developed: selective matching is done only considering the best $n$ solutions in DF and BF during a generation, so that worst routes are excluded *a priori*. A value $n = \frac{dim}{5}$, where *dim* is the problem dimension, i.e. the number of cities for TSP and VRP, was used in the simulations presented in this paper. The last strategy implemented was the *best-improved selective-matching*, where the $n$ best decisions in DF and BF are matched with the best global solution found by the algorithm at that time. A mix of these two last strategies (*improved selective-matching* followed by *best-improved selective-matching*, called *mix*M) was used for the test cases Eil51, Eil76, Eil101, Rat195, C50, C75, C100, C199, see Sect. 3.3. Strategy *n*M, meaning *no-matching*, will be used when no matching strategies are selected, that is when the *Physarums* solver has no communication ability and no matched paths are possible.

*Restart procedure.* A restart procedure was added in the algorithm to avoid stagnation at local minima. If a certain condition, here called *restart condition*, is reached, the veins' radii are reset to:

$$r_{ij} = R(r_{ij}) \tag{7}$$

where $R$ is a monotonic function of current radii, called restart function in the following. The first restart procedure introduced, called *restart1*, was based on the adaptive control of $GF$. This control was chosen in order to incrementally boost the effect of $GF$ during a simulation, driving exploring agents towards best veins. Simulations, see Sect. 3.3, showed that the adaptive control of $GF$ helps the convergence of the algorithm towards optimal solution when applied to *Ulysses16* and *Tuscany10* test cases. Given an initial value for the growth factor $GF_{ini}$, $GF$ is incremented by a fixed percentage $\sigma$ after every generation. If the probability $p_{best}$ associated with the best path so far (see adjacency probability matrix, Eq. (4)) is higher than a fixed value $p_{lim}^{low}$ , the increment is set to zero. Then, if $p_{best}$ exceeds a value $p_{lim}^{high}$ , $GF$ is set equal to $GF_{ini}$ and veins are dilated and contracted to their initial value. In the present paper it is assumed that $\sigma = 0.01$, $p_{lim}^{low} = 10^{-4}$ and $p_{lim}^{high} = 0.85$. This restart condition is not suitable for large scale problems (i.e. TSP and VRP with a number of cities greater than 20) because, due the mathematical nature of the algorithm coupled with the reinforced effect of $GF$, the search space is too rapidly reduced. For this reason, a second restart procedure, called *restart2*, was implemented. It is based on the number of nodes in common among decision sequences in a generation: after comparing all decision sequences among each other, if the minimum number of nodes in common $n_{min}^{com}$ exceeds a threshold $n_{com}$, the algorithm is restarted.

The main parameters of the modified *Physarum* solver are listed in Table 1. The initial radius of the veins $r_{ini}$ is always set equal to 1 in the simulations presented in this paper. The pseudocode of the multidirectional incremental modified *Physarum* solver is provided in Algorithm 1. The unidirectional algorithm is a special case of multidirectional algorithm, obtained by freezing the *BF*, i.e. flux and graph growth are allowed in only one direction.

### *2.4. Implementation styles*

The set of Eqs. (1)-(4) can be implemented following two different methods (or styles as they are called in the following). According to the first method, a mass balance in each node $i$ of the

graph is applied: assuming that radii, lengths, source (start node) and sink (end node) are known, fluxes can be computed solving the linear system generated from the continuity equations:

$$\begin{cases} \sum_j Q_{ij} = 0 \\ \sum_j Q_{Sj} - I_0 = 0 \\ \sum_j Q_{sj} + I_0 = 0 \end{cases} \tag{8}$$

where nodes $j$ are the neighbors of nodes $i$, $S$ is the source node, $s$ the sink node and $I_0$ the source term, i.e. the amount of blood pumped in the veins by the *Physarum*. Pressures can be then calculated via the Hagen-Poiseuille law and radii change with a fixed adaptation law, say Eq. (2) and Eq. (3). This method will be called fluid-style method. Applications of this method can be found in [24, 23]. In TSP and VRP problems, start node can be considered the node related to starting city acting like a source while end node can be a ghost node containing starting city information but acting as a sink.
The second method, the one adopted in this paper, was previously proposed in [13] in a similar fashion and resembles classical Ant Colony Optimization algorithms. This method will be called ACO-style. In accordance to Eq. (1), flux in each vein is proportional to the fourth power of radius and inversely proportional to the length. Once a vein is selected by a virtual agent in a generation, its radius is incremented using Eq. (2). In the present work, a function linear with respect to the product between the radius $r_{ij}^{(n)}$ of the veins traversed by agent $n$, and the inverse of the total cost of the decision taken by agent $n$, i.e. the total length $L_{tot}^{(n)}$, will be used for the veins' dilation:

$$\left.\frac{d}{dt} r_{ij}^{(n)}\right|_{dilation} = m \frac{r_{ij}^{(n)}}{L_{tot}^{(n)}} \tag{9}$$

where the coefficient $m$ is here called linear dilation coefficient. Evaporation is taken into account using Eq. (3) for each agent. Fluxes are then calculated using Eq. (1) and probabilities are updated in accordance with Eq. (4). Due to the mathematical nature of the algorithm (the flux is related to the fourth power of the radius), an upper limit on the maximum vein radius was introduced in order to avoid veins' flux explosion. If the radius $r_{ij}$ exceeds a maximum value $r_{max}$, the vein dilation is blocked up until the radius is again below $r_{max}$ for the effect of evaporation. This upper limit, called $k_{explosion}$, is given as ratio between $r_{ij}$ and $r_{ini}$:

$$k_{explosion} = \frac{r_{ij}}{r_{ini}} \tag{10}$$

where $r_{ini}$ is the initial radius of the veins.

In [13] there is a brilliant comparison between *Physarum* and ants algorithmic. The algorithm that reproduces the behaviour of starving ants looking for food appears in close analogy with the *Physarum* basic heuristic. The mechanism of deposition and evaporation of pheromone is mathematically comparable with the dilation and contraction processes of *pseudopodia*.
The first method seems to be very close to the physics of a real vein network, but a number of numerical experiments showed that the ACO-style performs much better. A comparison between the two styles will be discussed in Sect. 3.3.

## 3. Performance Evaluation

Traveling Salesman Problem (TSP) and Vehicle Routing Problem (VRP) are classical problems in combinatorial optimization and were used as benchmark for the performance evaluation


Figure 4: Map of Tuscany, Italy. *Tuscany10* VRP test case. Black circles are the selected cities, black square is the depot. Here is represented the best solution, calculated using an exhaustive search, for the case $n = 9, k = cost = 1, v = 1, c = 4, d = 1$. Map courtesy of *Google maps*.

of the modified unidirectional and multidirectional *Physarum* algorithm.

### *3.1. Benchmark*

Given a set $S\{n, l(n)\}$ of $n$ cities whose reciprocal distance $l$ is known, the TSP is the problem of finding the shortest tour that visits each city exactly once. The problem is mathematically treated in graph theory under the name of Hamiltonian path problem. We will refer exclusively to symmetric TSP, namely undirected graphs. Problems reducible to TSP have a lot of applications in various fields. Examples are logistic, planning and DNA sequencing.

VRP is a similar problem. Given a set $S\{n, l(n), k(n), v, c(v), d, l_d(n, d)\}$ of $n$ cities with a demand $k$, whose reciprocal distance $l$ is known, $v$ vehicles of capacity $c$, $d$ depots and the distance $l_d$ between depots and cities, the VRP is the problem of delivering goods located in fixed depots using a defined amount of vehicles with finite capacity. The goal is to satisfy the demand of each city minimizing the costs, i.e. the distance. VRP reduces to a TSP if there is only one vehicle with infinite capacity.

When expressed as decision problems, TSP and VRP are NP-complete. Today no exact algorithms able to solve NP-complete problems using an acceptable amount of time independently from the dimension has been discovered. Referring to TSP made of $n$ cities, the running time of exhaustive search is O($n!$). Bio-inspired computing offers the possibility of finding quickly good solutions for problems of increasing dimensions. It is of interest the fact that a good percentage of humans are able to draw near-optimal solutions for TSP problems with a number of cities of the order of tens [6].

TSPLIB [26] was used to benchmark the proposed *Physarum* algorithm, developed in Matlab® R2010b, on the TSP problem. Sect. 3.3 reports the results obtained by applying the multidirectional solver to test cases Ulysses16, Eil51, Eil76, Eil101 and Rat195. The reference optimal solutions and factors of normalisation used for these particular TSP instances are shown in Tab. 2.

The multidirectional modified *Physarum* solver applied to VRP was initially tested on a map of 9 cities plus one depot. The map is built using 9 Italian cities (Firenze, Livorno, Montecatini, Pistoia, Prato, Montevarchi, Arezzo, Siena, San Gimignano), with a city considered the depot

Table 2: TSP & VRP Test cases best-known solutions, exact and normalised.

| Test Case | Type | Best-known, exact | Best-known, normalised | Factor of normalisation |
|---|---|---|---|---|
| Ulysses16 | TSP | 6859 | 0.6859 | 10000 |
| Eil51 | TSP | 426 | 0.4260 | 1000 |
| Eil76 | TSP | 538 | 0.3587 | 1500 |
| Eil101 | TSP | 629 | 0.2690 | 2000 |
| Rat195 | TSP | 2323 | 0.4224 | 5500 |
| Tuscany10 | VRP | 400 | 400 | 1 |
| C50 | VRP | 524.61 | 0.5246 | 1000 |
| C75 | VRP | 835.26 | 0.4176 | 2000 |
| C100 | VRP | 826.14 | 0.2065 | 4000 |
| C199 | VRP | 1291.45 | 0.2583 | 5000 |

(Ponsacco, see Fig. 4). The Euclidean distance in kilometers was used. Optimal tour with parameters $n = 9, k = cost = 1, v = 1, c = 4, d = 1$ was found using an exhaustive search and is shown in Fig. 4. This instance will be called *Tuscany10* in the following. Then, the algorithm was tested on the VRP instances C50, C75, C100 and C199, as in [7]. The reference optimal solutions and factors of normalisation used for these particular VRP instances are shown in Tab. 2. Results are shown in Sect. 3.3.

### *3.2. Testing procedure*

The testing procedure proposed in [27] was used in this paper. The index of performance $j_s$ used to explore the efficiency of the algorithm and to compare the multidirectional and the unidirectional versions, is the number of success of the algorithm, here called $P$. The algorithm is considered to be successful when the best computed value $f^{best}$ is lower than the best known solution $f_p^{best}$ for a given problem $p$, plus a tolerance $tol$.

**Algorithm 2** Testing procedure

```
set to N the max number of function evaluations for P
apply P to p for n times and set j_s = 0
for all i ∈ [1, ..., n] do
   if f_i^best ≤ f_p^best + tol then
      j_s = j_s + 1
   end if
end for
evaluate p_s = j_s/n
```

The success rate $p_s = j_s/n$, expressing the percentage of success after $N$ function evaluations of $P$ over a group of $n$ simulations, will be used for the comparative assessment of the algorithm's performance. A function evaluation is defined as the call to the objective function, i.e. each arc selected by the virtual exploring agents (see Sect. 2) is considered a function evaluation. For

Table 3: Tolerances for success rate calculation express as percentage of best known values.

| Test Case | Type | Tolerance |
|---|---|---|
| Ulysses16 | TSP | $10^{-8}$% |
| Eil51 | TSP | 0.25% |
| Eil76 | TSP | 0.25% |
| Eil101 | TSP | 2% |
| Rat195 | TSP | 3% |
| Tuscany10 | VRP | $10^{-8}$% |
| C50 | VRP | 7% |
| C75 | VRP | 15% |
| C100 | VRP | 21% |
| C199 | VRP | 25% |

the unidirectional algorithm, during each generation the number of calls for a VRP problem is $N_{calls} = \left(n_{city} + n_{returns}\right) n_{agents} + B_{arcs}$, where $n_{city}$ is the number of cities to visit, $n_{returns}$ the number of returns to the depot and $n_{agents}$ the number of exploring agents and $B_{arcs}$ the number of new arcs built. For the multidirectional algorithm, during each generation the number of calls for a VRP problem is $N_{calls} = 2\left(n_{city} + n_{returns}\right) n_{agents} + B_{arcs}$. For a TSP problem $n_{returns} = 0$. The overall process leads to Algorithm 2. In [27] one can find a full explanation on the setting of the value of $n$ in order to have a reliable estimate of algorithm's success probability. A value equals to 200 for $n$ is used in the simulations presented in this paper. Tolerances for each test case are reported in Tab. 3.

### *3.3. Results*

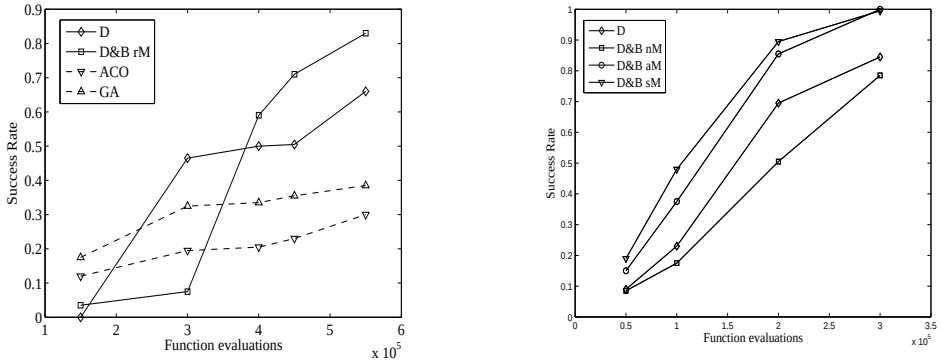
The multidirectional modified *Physarum* solver, named D&B in the following, was compared against a unidirectional modified *Physarum* solver, named D. First of all the two styles (ACO-style and fluid-style, introduced in Sect. 2.4) were tested on a simple Traveling Salesman Problem made of 14 cities placed on a circumference of radius equals to 50 and with identical relative angle. Experimental results showed that when the ACO-style algorithm achieved a success rate of 95%, the fluid-style did not exceed the 10% at same function evaluations. For this reason, the ACO-style was selected in this paper.

*Ulysses16 & Tuscany10 test cases.* The multidirectional and unidirectional modified *Physarum* algorithms were applied first to the symmetric TSP test case Ulysses16 and VRP test case Tuscany10, described in Sect. 3.1. Simulations on these test cases were carried out using the restart procedure *restart1*, based on the adaptive control of the growth factor $GF$, discussed in Sect. 2. The restart function $R$ (see Eq. (7)) and the vein ratio limit (see Eq. (10)) chosen were $R(r_{ij}) = r_{ini}$ and $k_{explosion} = 10^6$. The values used as setting parameters in the simulations for the Ulysses16 are $m = 5 \times 10^{-5}$, $\rho = 1 \times 10^{-5}$, $GF_{ini} = 5 \times 10^{-3}$, $N_{agents} = 100$, $p_{ram} = 0.9$, $\alpha = 0$, while for the Tuscany10 are $m = 1 \times 10^{-5}$, $\rho = 1 \times 10^{-5}$, $GF_{ini} = 5 \times 10^{-3}$, $N_{agents} = 150$, $p_{ram} = 0.9$, $\alpha = 0$, and were chosen after a series of experimental tests.

After analyzing the results on Ulysses16 shown in Fig. 5(a), one can observe that the multidirectional modified *Physarum* algorithm with matching ability (D&B with rM) provides higher

Table 4: Mean saving in computational time [%], over 200 runs at 1, 2, $3 \cdot 10^5$ function evaluations, using D&B with sM instead of D&B with aM.

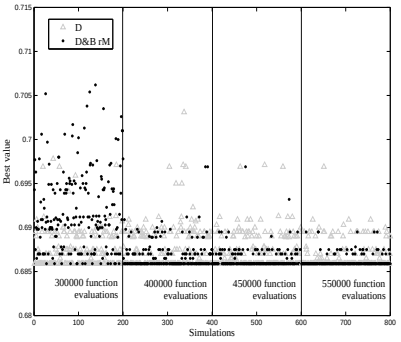
| Evaluations | $1 \cdot 10^5$ | $2 \cdot 10^5$ | $3 \cdot 10^5$ |
|---|---|---|---|
| sM vs. aM | 33% | 34% | 37% |



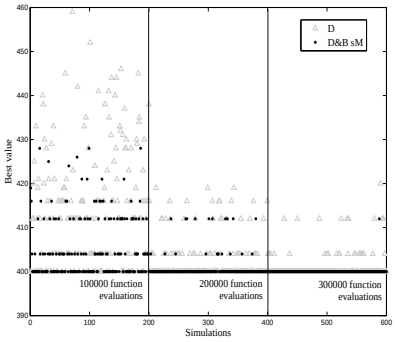

(a) Ulysses16 (b) Tuscany10

Figure 5: Variation of the success rate with the number of function evaluations - Ulysses16 & Tuscany10 test cases.

success rate than the unidirectional modified *Physarum* algorithm (D). Although D performs better at $3 \cdot 10^5$ function evaluations (success rate is near 0.5 while success rate of D&B with rM is under 0.1), the performance of multidirectional exceeds the performance of direct only solver increasing the number of function evaluations. At $5.5 \cdot 10^5$ function evaluations the success rate of D&B with rM reaches 0.83, against the 0.66 of D. Fig. 5(a) shows also the performance of a simple Ant Colony Optimization (ACO, [28]) and genetic algorithm (GA, Matlab® R2010b global optimization toolbox) solvers. The ACO solver is the implementation of the algorithm provided in [11]. Results at $5.5 \cdot 10^5$ function evaluations show that D&B and D&B with rM have a good performance (success rate are respectively 0.66 and 0.83) if compared with a basic ACO solver (success rate is 0.30) and GA solver (success rate is 0.38). This comparison was made in order to check if the required function evaluations for the *Physarum* solver were comparable, in terms of order of magnitude, with a basic ACO solver. A comparison with more advanced implementations of ACO is provided in Sect. 3.6. Fig. 6(a) shows a comparison among the best solutions for each of the 200 runs at 300000, 400000, 450000, 550000 function evaluations. The best solution is known to be 0.6859 (TSPLIB [26]) for the TSP test case (see Sect. 3.1). As the number of function evluations increases, the D&B with rM algorithm (black dots) tend to have an output comprised in a narrower band of values if compared with D algorithm (grey dots). At $3 \cdot 10^5$ function evaluations there is a 9% probability to reach the global minimum value, a 71% probability to reach a value in the band $(0.6859, 0.6978]$ and 20% probability to reach a value in the band $(0.6978, 0.7070]$ using D&B with rM, while, using D, there is a 48% probability to reach the exact value, a 52% to reach a value between $(0.6859, 0.6978]$ and a 0% probability to reach




(a) Ulysses16 - D and D&B with rM - best value = 0.6859




(b) Tuscany10 - D and D&B with sM - best value = 400

Figure 6: Best solution found by D and D&B on groups of 200 runs for different function evaluation limits - Ulysses16 & Tuscany10 test cases.

a value in the band $(0.6978, 0.7070]$. At $5.5 \cdot 10^5$ function evaluations the probability of exact value outcome using D&B with rM rise up to 83% with a 17% probability to reach a value in the band $(0.6859, 0.6950]$, while there is only a 66% probability using D, with a 27% probability to reach a value in the band $(0.6859, 0.6950]$ and a 7% in the band $(0.6950, 0.6978]$.

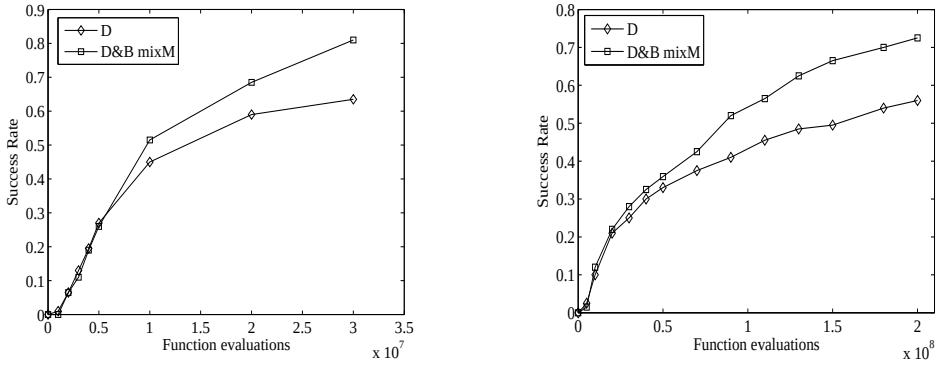
As for Ulysses16, Tuscany10's results, shown in Fig. 5(b), demonstrate that the multidirectional algorithm with matching ability (D&B with *s*M and D&B with *a*M) performs better than the unidirectional algorithm (D). After analysing more in depth the results and considering the mathematical modeling in Sect. 2, one could argue that:

I. At 300000 function evaluations the gain in success rate using D&B with *s*M or *a*M instead of D is around 20%. As in the *Ulysses16* test case, this gain is due to the forward and backward propagation of decision sequences enhanced by a communication ability (matching): *pseudopodia* can evaluate each step of the decision sequence from two directions, and can create joint decision sequences.

II. The performance of D&B with *s*M and D&B with *a*M are almost comparable. The real gain using D&B with *s*M is the saving in computational time. The mean computational time in the simulations (200 runs) is reduced considerably (between 33% and 37%) using a selective matching instead of a non selective matching. This is due to the selection of best joint decision sequence at each generation: discarding worst joint decision sequences the updating of veins is done only for the best sequences. Tab. 4 shows the comparison.

III. The multidirectional algorithm without matching ability (D&B with *n*M) has the worst performance compared to others in all the cases. This is due to lack of communications between the two *Physarum*. These data demonstrate that a matching condition is necessary for an optimal working of multidirectional algorithm.

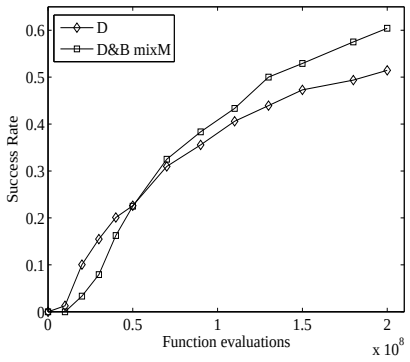
Fig. 6(b) shows a comparison among the best solutions for each of the 200 runs, using D and D&B with sM, at 100000, 200000 and 300000 function evaluations. The best solution was proved to be 400, after an exhaustive search, for the VRP test case (see Sect. 3.1). The figure shows that the D&B with sM algorithm (black dots) has an output comprised in a narrower band of values if compared with the D algorithm (grey dots). At $1 \cdot 10^5$ function evaluations there is



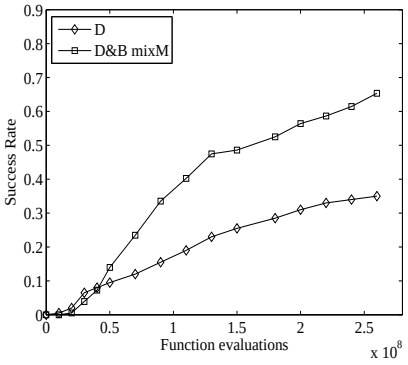

(a) Eil51




(b) Eil76




(c) Eil101




(d) Rat195

Figure 7: Variation of the success rate with the number of function evaluations - Eil51, Eil76, Eil101 & Rat195 TSP test cases.

about a 50% probability to reach the exact value and a 50% probability to reach a value in the interval $(400, 428]$ using D&B with sM, while, using D, there is a 23% probability to reach the exact value, a 44% probability to reach a value in $(400, 428]$ and a 33% to reach a value in the interval $(428, 459]$. At $3 \cdot 10^5$ function evaluations the probability of an exact value outcome using D&B with sM rise up to 99%, while there is only a 84% probability using D, with a 16% probability to reach a value in the interval $(400, 428]$.

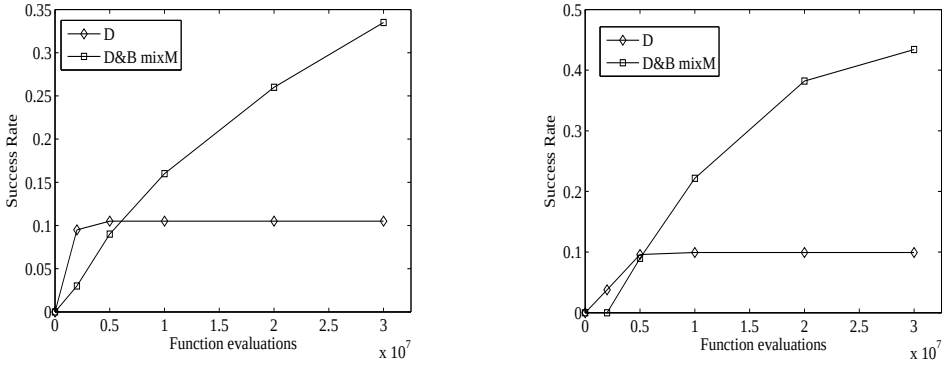
*TSP Eil51, Eil76, Eil101, Rat195 test cases.* Fig. 7 shows a comparison between the multidirectional and the unidirectional modified *Physarum* solver for the TSP test cases Eil51, Eil76, Eil101 & Rat195. Setting values are $m = 5 \cdot 10^{-3}$, $\rho = 10^{-4}$, $GF = 5 \cdot 10^{-3}$, $N_{agents} = 100$, $p_{ram} = 1$, $\alpha = 0$. The value $k_{explosion} = 5$, see Eq. (10) and the *Restart2* with $n_{com} = \frac{dim}{2}$ were selected, where *dim* is the problem dimension, i.e. the number of cities. The matching strategy is the *mix*M. Results in Fig. 7 demonstrate that the multidirectional modified *Physarum* algorithm with matching ability (D&B with *mix*M) provides higher success rate than the

unidirectional modified *Physarum* algorithm (D) when applied to larger scale TSP problems. In particular for Eil51, Fig. 7(a), and Eil76, Fig. 7(b), the gain using D&B with *mix*M instead of D reaches approximately 20% respectively at $3 \cdot 10^7$ and $2 \cdot 10^8$ function evaluations. For Eil101 at $2 \cdot 10^8$ function evaluations, Fig. 7(c), this gain results to be around 10%, while for Rat195 at $2.5 \cdot 10^8$ function evaluations, Fig. 7(d), it is 30%. The possibility of evaluating decision sequences from multiple directions and the ability of communication between the counter expanding *Physarum* lead to an improvement in the algorithm performance, i.e. the success rate vs. function evaluations, and this gain is here up to 30%. It should be noted that for the TSP instances presented in this paragraph, there is an initial transient phase where the performance of the multidirectional solver do not exceed the performance of the unidirectional solver. This behaviour can be interpreted as an initial inertia of the multidirectional algorithm. The two expanding *Physarum* double the calls to the objective function, causing an initial phase where the number of function evaluations is high while the growth of the veins is still low. After this initial transient, the two *Physarum* start to reduce the search space and the role of communication becomes crucial. The result is a fast performace increment after this start-up phase.

*VRP C50, C75, C100, C199 test cases.* The modified *Physarum* algorithm was applied also to the VRP test cases C50, C75, C100, C199. Setting values are $m = 10^{-2}$, $\rho = 10^{-4}$, $GF = 5 \cdot 10^{-3}$, $N_{agents} = 100$, $p_{ram} = 1$, $\alpha = 0$, the matching strategy used is the *mix*M and the restart procedure is the *restart2* with $n_{com} = \frac{dim}{8}$, where *dim* is the problem dimension, i.e. the number of cities. A principal square root function was chosen for the restart function, see Eq. (7), in order to preserve some acquired information on the search space after restart, and $k_{explosion} = 5$, see Eq. (10), was selected. Fig. 8 shows a comparison between the multidirectional and the unidirectional modified *Physarum* solver for these test cases. Results for C50, C75 and C100 in Fig. 8(a)-(c) show an initial transient phase (function evaluations under $5 \cdot 10^6$) where the unidirectional algorithm provides a higher, although low (around 10% for C50 and C75, 30% for C100), success rate. After this initial transient, the multidirectional algorithm's performance exceeds the one of the unidirectional algorithm: the gain in terms of success rate for the C50 and C75 at $3 \cdot 10^7$ function evaluations is approximately 35%, while the gain for C100 reaches 20% at $2.5 \cdot 10^7$ function evaluations. For the C199 test case, Fig. 8(d) the initial transient phase is slightly longer ($1.5 \cdot 10^7$ function evaluations), with the D&B success rate achieving 45% at $2.5 \cdot 10^7$ function evaluations while D do not exceed 25% at same function evaluations. As for the larger scale TSP test cases and for Ulisses16 and Tuscany10, these results on larger scale VRP test cases indicates that the introduction of multidirectionality coupled with a matching strategy is an optimal choice if compared with the classical unidirectional approach. The considerations for the start-up phase in the previous paragraph apply here as well.

### *3.4. Parameter Tuning*

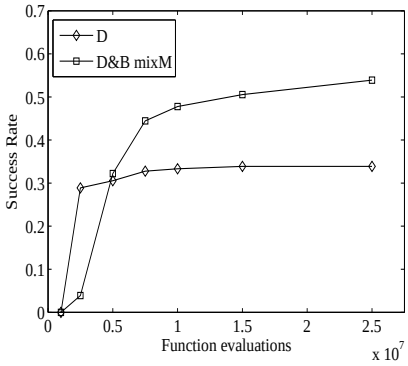
Fig. 9 shows the effect of the variation of some key parameters on the success rate for the C199 test case. The difference in success rate is given as percentage of variation with respect to the values reported in Fig. 8(d). The key parameters chosen are the linear dilation coefficient $m$, the growth factor $GF$, the radius upper limit $k_{explosion}$ and the restart threshold $n_{com}$, introduced above. Starting from the base setting values $m = 10^{-2}$, $\rho = 10^{-4}$, $GF = 5 \cdot 10^{-3}$, $N_{agents} = 100$, $p_{ram} = 1$ and $\alpha = 0$ used for the C50 to C199 problems, the key parameters are varied while keeping the others constant. Analyzing Fig. 9, one could argue that:



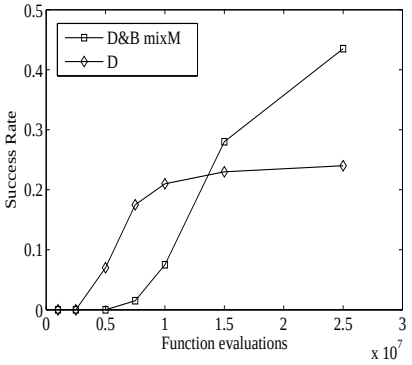

(a) C50




(b) C75




(c) C100

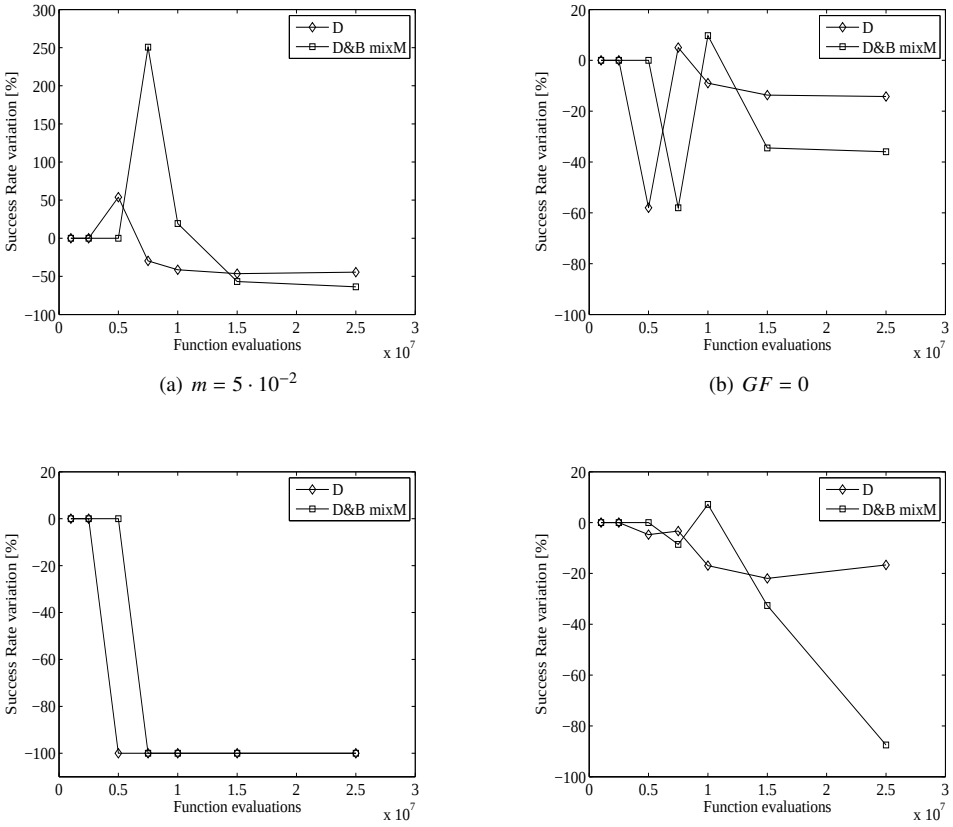



(d) C199

Figure 8: Variation of the success rate with the number of function evaluations - C50, C75, C100 & C199 VRP test cases.

I. Fig. 9(a) shows that both the unidirectional and multidirectional algorithms are sensitive to small variation of the linear dilation coefficient $m$. A high value of $m$ causes a fast growth of the veins, the exploring agents tend to choose a limited set of decisions quickly and the veins reach the upper radius limit after few generations. The result is an increase in the success rate in the early stage of the simulation, with a decrease of success rate up to 65% at $25 \cdot 10^6$ function evaluations for the multidirectional algorithm and up to 45% at $25 \cdot 10^6$ function evaluations for the unidirectional algorithm.The initial increased performance is up to 250% for the multidirectional *Physarum* solver at $7.5 \cdot 10^6$ function evaluations. On the other hand, a low value of $m$ would lead to a slow growth of the veins, meaning that the search space is reduced slowly during the simulation, with a decrease of performance.

II. Fig. 9(b) shows that the introduction of the growth factor in the algorithm seems to have a positive effect. If the *GF* is set to zero, the performance of the *Physarum* solver is reduced by up to 60% during the early stage of simulations for both the multidirectional and unidirectional algorithms. After that, there is an oscillation that leads to a small gain in success rate (at $5 \cdot 10^6$




(a) $m = 5 \cdot 10^{-2}$ (b) $GF = 0$

(c) $k_{explosion} = 10$ (d) $n_{com} = 80$

Figure 9: Variation of success rate with key parameters - C199 VRP test case.

evaluations for the unidirectional algorithm and at $7.5 \cdot 10^6$ evaluations for the multidirectional solver), with a reduction of success rate increasing the number of function evaluations. This reduction is up to 40% for the D&B solver and 15% for the D solver at $25 \cdot 10^6$ function evaluations. It should be noted that, on the other hand, a high value of $GF$ would cause a too fast increment of the best veins' radii, causing an accumulation of exploring agents in the neighborhood of the best solution found after few generations. This would lead to a decrease of performance.

III. Fig. 9(c) shows that the algorithms seem to be very sensitive to the radius upper limit $k_{explosion}$: its introduction is vital when the *Physarum* algorithm is applied to large-scale problems. An increase in this upper limit causes a fall of success rate to 0% for both the unidirectional and multidirectional algorithm.

IV. Fig. 9(d) shows that the restart condition based on nodes in common, introduced above for the large-scale problems, affects the performance. A higher value of the restart threshold $n_{com}$ causes a smaller number of restarts of the algorithm in time with a decrease of performance of up to 90% for the multidirectional algorithm at $25 \cdot 10^6$ function evaluations and up to 20% for the unidirectional algorithm at $25 \cdot 10^6$ function evaluations.

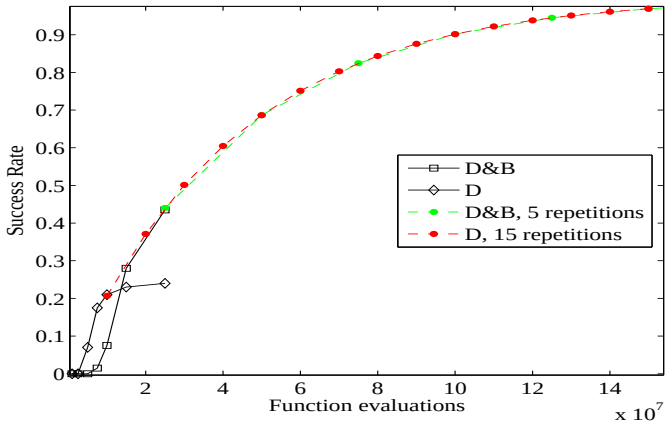



Figure 10: Extrapolated probability of success - C199 VRP test case.

### *3.5. Extrapolated Probability of Success*

It is of interest the calculation of the success rate after $n$ independent runs of the algorithm. Starting from the success rate at a given number of function evaluations, it is possible to extrapolate the probability of success $P_s$ of the algorithm after $n$ independent runs, each with an equal number of function evaluations:

$$P_s^{(n)} = 1 - (1 - p_s^*)^n \tag{11}$$

where $p_s^*$ is the success rate at a chosen function evaluation value from simulation data and $(1 - p_s^*)^n$ is the probability of insuccess of the algorithm after $n$ repetition. The value of $P_s$ is here equals to the success rate $p_s$. Fig. 10 reports the probability of success of the unidirectional and multidirectional *Physarum* solvers after $n$ repeated runs. Each dot represents a repetition of the algorithm. The probability of success of D only solver (red line) is calculated from the success rate value (21%) at $10^7$ function evaluations, while the probability of success of D&B solver (green line) is calculated from the success rate value (44%) at $2.5 \cdot 10^7$ function evaluations. Fig. 10 shows that the two extrapolated curves are almost identical, meaning that, for the C199 test case, the multidirectional and the unidirectional approaches are identical in terms of success rate if the algorithm runs are properly repeated: this would imply the necessity of algorithm supervision, that makes again the unidirectional solver less attractive than the multidirectional one. Furthermore, it should be noted that the performance of the multidirectional algorithm may increase significantly with new matching and restart strategies. This will be further investigated in the future.

### *3.6. Comparison with other algorithms*

Although the idea of this work is to prove that multidirectionality increases the performance of the *Physarum* algorithm, in this section a comparison with known and tested algorithms is

Table 5: Comparison - TSP instances Eil51, Kroa100, d198 (part of the benchmark used in [10]). Mean, standard deviation and best (rows) are calculated at approximately $(3, 10, 40)\cdot 10^7$ calls to the objective function for all the algorithms (*2-opt* calls not counted).

| | Physarum D&B *mix*M | MMAS | ACS + cl | AS | $AS_e$ | $AS_r$ | $AS_r$ + pts |
|---|---|---|---|---|---|---|---|
| *Eil51* | | | | | | | |
| mean | 426.8 | 427.2 | 428.1 | 437.3 | 428.3 | 434.5 | 428.8 |
| variance | 0.92 | 1.13 | 2.48 | n/a | n/a | n/a | n/a |
| best | 426 | 426 | 426 | n/a | n/a | n/a | n/a |
| *Kroa100* | | | | | | | |
| mean | 21352.9 | 21352.1 | 21420.0 | 22471.4 | 21522.8 | 21746.0 | 21394.9 |
| variance | 94.7 | 50.3 | 141.7 | n/a | n/a | n/a | n/a |
| best | 21282 | 21282 | 21282 | n/a | n/a | n/a | n/a |
| *d198* | | | | | | | |
| mean | 16461.0 | 16066.0 | 16054.0 | 16702.1 | 16205 | 16199.1 | 16025.2 |
| variance | 235.2 | 73.82 | 71.15 | n/a | n/a | n/a | n/a |
| best | 15942 | 15960 | 15888 | n/a | n/a | n/a | n/a |

ACS is from [10], 15 runs performed.
MMAS is from [18], 25 runs performed.
AS and variants are from [19], 25 runs performed.
Physarum, 25 runs performed for *Kroa100* and *d198*, 200 for Eil51.

Table 6: Comparison - TSP instances Eil51,Eil76, Eil101, Rat195 (part of the benchmark used in [30]). Mean is calculated at approximately $(1, 2, 4, 15)\cdot 10^7$ calls to the objective function for all the algorithms.

| | Physarum D&B *mix*M | KniesG | KniesL | SA |
|---|---|---|---|---|
| *Eil51* | 427.8 | 438.2 | 438.2 | 435.9 |
| *Eil76* | 543.7 | 567.5 | 564.8 | 567.8 |
| *Eil101* | 649.8 | 664.4 | 658.3 | 665.1 |
| *Rat195* | 2397.0 | 2599.9 | 26073.0 | 2631.7 |

KniesG, KniesL, SA are from [30], runs performed n/a.
Physarum, 200 runs performed.

proposed in order to show strengths and weakness of the D&B with *mix*M. Only problems with more than 50 cities will be considered in this comparison. It is important to bear in mind that the *Physarum* algorithm, in its basic form, is devised to solve generic discrete decision making problems and does not contain either tailored heuristics for TSP and VRP [4] or local optimisation heuristics, like *2-opt*, *2.5-opt* and *3-opt* (widely used in combinatorial optimisation to improve TSP and VRP solutions [31, 5]), or static or dynamic candidate lists [18, 19, 10]. The use of candidate lists is problem specific and a heuristic that performs optimally on TSP and VRP may not be ideal for other problems.

*ACS, MMAS, AS, KniesG, KniesL, SA for TSP*. For this set of comparative tests, mean, best and standard deviation are used as performance indicators, see Table 5 and Table 6, in order to have a fair comparison with the algorithms in the literature.

Values in Table 5 for the ACO-inspired algorithms are from [10, 18, 19]. AS refers to Ant Systems, ACS to Ant Colony System, a first improvement of AS, MMAS to Max-Min Ant System, a further improvement which is considered the state of the art of ACO-inspired algorithms. The subscripts *e* and *r* are used to indicate Elitist and Rank-based Ant Systems repectively. +cl is added when candidate list is used, while +pts indicates the pheromone trail smoothing mechanism. Table 5 shows that the proposed *Physarum* algorithm is able to find good solutions if compared with aco-inspired algorithms. For the *Eil51* case the mean value and standard deviation found by the *Physarum* are the best one. For *Kroa100* the *Physarum* performs better than all the other algorithms in terms of mean value, except MMAS that achieve approximately the same mean value with a better standard deviation. Increasing the dimension of the problem to 198, the *Physarum* seems less performing than other algorithm (except AS), although its mean value is only 2.5% higher than MMAS that find a worse best value (15942 vs. 15960). The poorer performance is due to the fact that the *Physarum* with multidirectional ability at $10^8$ and $4 \cdot 10^8$ function evaluations is still in its start-up transient phase, see Sect. 3.3: increasing the number of function evaluations would increase the performance with a low probability of getting trapped in local minima.

Values in Table 6 are from [30]. KniesG refers to Kohonen Network Incorporating Explicit Statistics Global [1], KniesL to its local version [1], while SA indicates Simulated Annealing [3]. Results demonstrate the superior ability of the *Physarum* algorithm in finding good solutions

Table 7: Comparison -VRP instance C50. Mean, standard deviation and best (rows) calculated at $10^7$ calls to the objective function for all the algorithms (*2-opt* calls not counted).

| | Physarum D&B *mix*M | Physarum D&B *mix*M+*2opt* | ACO +cl+*2opt* | $ACO_m$ +cl+*2opt* |
|---|---|---|---|---|
| *C50* | | | | |
| mean | 571.6 | 538.24 | 552.0 | 550.1 |
| variance | 11.9 | 7.6 | 3.84 | 3.30 |
| best | 545.0 | 524.9 | 540.3 | 535.2 |

ACOs are from [5], runs performed 25.

for all the test cases *Eil51*, *Eil76*, *Eil101* and *Rat195*. [30] presents also an algorithm, called ACO&PR, that mixes ACO, greedy heuristics and path relinking with *2-opt* applied to each tour built. However, the number of calls per iteration is expected to be considerably high, as remarked also in [30], and is not reported in [30]. Therefore, it cannot be compared to the results of the *Physarum* solver.

*ACO variants for VRP*. A comparison on the VRP against other aco-inspired algorithms is tricky because most of the algorithms in literature make use of local optimisation heuristics such as *2-opt* and problem specific improvements. Table 7 shows a comparison with some ACO implementations in [5]. The subscprit *m* stands for Multiple ACO. These algorithms are provided with *2-opt* and candidate list. The table reports values with bigger candidate list of size $dim/3$, where *dim* is the problem dimension. Results show that the performance of the pure *Physarum* algorithm is not as good as the other two ACO algorithms, although the maximum difference is only 3.76% in the mean value and 2% in the best value. This comparison is, however, not completely fair as the number of function evaluations does not include the function calls required by the *2-opt* heuristics. If the *2-opt* heuristics is plugged into the *Physarum* algorithm, see Table 7 second column, its performance exceeds the one of the other ACO algorithms both in terms of mean and best value. Fig. 11 shows the variation in success rate when adding the *2-opt* heuristics. Furthermore, as shown in [4], ACO-inspired algorithms, with *2-opt* heuristic, can further increase their performances by incorporating heuristics that are problem specific.

Given the similarities between ACOs and *Physarum*, see Sect. 2.4, and considering the results achieved adding *2-opt*, it is reasonable to assume that the *Physarum* algorithm would further improve its performance on VRP with problem specific heuristics already developed for ACO algorithms. Testing these improvements, however, is beyond of the scope of this work, which aims at demonstrating that the introduction of multidirectionality is an optimal choice for the *Physarum* solver.

## 4. Conclusion

This paper introduced an innovative algorithm for optimal single objective discrete decision making, that draws inspiration from the behaviour of the slime mould *Physarum Polycephalum*

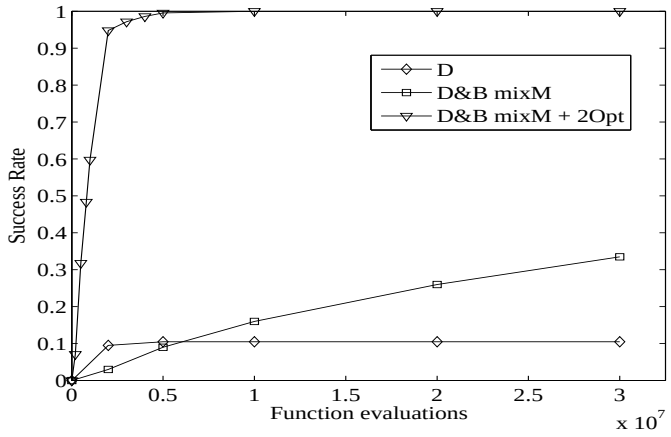



Figure 11: Success rate improvement adding 2-opt heuristic with $R(r_{ij}) = r_{ini}$ (see Eq. (7)) - C50 VRP test case. 2-opt exchanges are not here considered calls to the objective function.

looking for food. The introduction of multidirectionality and matching abilities (D&B with mixM) in the construction of decision sequences was demonstrated to be an optimal choice if compared with the classical unidirectional approach in terms of success rate for a given number of calls to the objective function.The possibility of building and exploring decision networks from multiple directions enhances the performances of the solver. The benchmark for algorithms' comparison was made of TSP and VRP instances with a number of cities between 10 and 199: in all the test cases the multidirectional modified *Physarum* solver with matching ability was able to achieve higher success rates than the unidirectional modified Physarum solver. Furthermore the multidirectional algorithm showed its competitiveness in solving TSP and VRP problems if compared with some state-of-the-art algorithms in the literature even without the use of local heuristics, problem specific improvements and candidate lists. It should be noted that the algorithm is able to grow and explore decision graphs from multiple directions. It will be part of the future work the exploration of optimal ways and laws for growing decision networks in multiple directions with a graph topology rather than a mesh topology.

## References

[1] Aras, N., Oommen, B.J., Altinel, I.K., 1999. The Kohonen Network Incorporating Explicit Statistics and its Application to the Traveling Salesman Problem. Neural Networks 12(9), 12731284.

[2] Adamatzky, A., Martinez, G.J., Chapa-Vergara, S.V., Asomoza-Palacio, R., Stephens, C.R., 2011. Approximating Mexican highways with Slime Mould. Natural Computing 10(3), 1195-1214.

[3] Budinich, M., 1996. A Self-organizing Neural Network for the Traveling Salesman Problem that is Competitive with Simulated Annealing. Neural Computation 8, 416424.

[4] Bullnheimer, B., Hartl, R. F., & Strauss, C., 1999. Applying the Ant System to the Vehicle Routing Problem. Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization. 109-120.

[5] Bell, J. E., McMullen, P. R., 2004. Ant Colony Optimization Techniques for the Vehicle Routing Problem. Advanced Engineering Informatics, 18(1), 41-48.

[6] Burns, N. R., Lee, M.D., Vickers, D., 2006. Are Individual Differences in Performance on Perceptual and Cognitive Optimization Problems Determined by General Intelligence?. The Journal of Problem Solving 1(1), 5-19.

[7] Bin Y., Zhong-Zhen Y., Baozhen Y., 2009. An Improved Ant Colony Optimization for Vehicle Routing Problem, European Journal of Operational Research 196. 171-176.
[8] Chiung Moon, Jongsoo Kim, Gyunghyun Choi, Yoonho Seo, 2002. An efficient genetic algorithm for the traveling salesman problem with precedence constraints. European Journal of Operational Research 140(3), 606-617.
[9] Chen, C.H., Ting, C.J., 2006. An Improved Ant Colony System Algorithm for the Vehicle Routing Problem. Journal of the Chinese Institute of Industrial Engineers 23(2), 115-126.
[10] Dorigo M., Gambardella L.M., 1996. Solving Symmetric and Asymmetric TSPs by Ant Colonies. Proceedings of IEEE International Conference on Evolutionary Computation ,622-627.
[11] Dorigo, M., Maniezzo V., Colorni A., 1996. The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B 26(1), 29-41.
[12] Dorigo M., Gambardella L.M., 1997. Ant Colonies for the Travelling Salesman Problem. BioSystems 43, 73-81.
[13] Hickey, D.S., Noriega, L.A., 2008. Insights into Information Processing by the Single Cell Slime Mold Physarum Polycephalum. 2008 UKACC Control Conference, Manchester University.
[14] Monismith Jr., D.R., Mayfield, B.E., 2008. Slime Mold as a Model for Numerical Optimization. 2008 IEEE Swarm Intelligence Symposium, St. Louis MO, USA.
[15] Morin S., Gagn, C., Gravel, M., 2009. Ant colony optimization with a specialized pheromone trail for the car-sequencing problem. European Journal of Operational Research 197(3), 1185-1191.
[16] Masi, L., Vasile, M., 2012. A Multidirectional Modified Physarum Solver for Optimal Discrete Decision Making. Proceedings of the International Conference on Bio-Inspired Optimization Methods and their Applications, BIOMA 2012, Bohinj, Slovenia.
[17] Nakagaki, T., Yamada, H., Toth, A., 2000. Maze-Solving by an Amoeboid Organism. Nature 407, 470.
[18] Stuetzle, T., Hoos, H., 1997. Max-Min Ant System and Local Search for the Traveling Salesman Problem. IEEE International Conference on Evolutionary Computation, 309-314.
[19] Stuetzle, T., Hoos, H., 2000. Max-Min Ant System. Future Generation Computer System, 16(8), 889-914.
[20] Szeto, W.Y., Yongzhong Wu, Sin C. Ho, 2011. An artificial bee colony algorithm for the capacitated vehicle routing problem. European Journal of Operational Research 215(1), 126-135.
[21] Tsai, Cheng-Fa, Chun-Wei Tsai, and Ching-Chang Tseng, 2004. A New Hybrid Heuristic Approach for Solving Large Traveling Salesman Problem. Information Sciences 166(1), 67-81.
[22] Tasgetiren, M. F., Yun-Chia Liang, Sevkli, M., Gencyilmaz, G., 2007. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. European Journal of Operational Research 177(3), 1930-1947.
[23] Tero, A., Kobayashi, R., Nakagaki, T., 2006. Physarum Solver: a Biologically Inspired Method of Road-Network Navigation. Physica: A Statistical Mechanics and its Applications 363(1), 115-119.
[24] Tero, A., Yumiki, K., Kobayashi, R., Saigusa, T., Nakagaki, T., 2008. Flow-Network Adaptation in Physarum Amoebae. Theory in Biosciences 127(2), 89-94.
[25] Tero, A., Takagi, S., Saigusa, T., Ito, K., Bebber, D.P., Fricker, M.D., Yumiki, K., Kobayashi, R., Nakagaki, T., 2010. Rules for Biologically Inspired Adaptive Network Design. Science 439, 327.
[26] TSPLIB, library of instances for travelling salesman and vehicle routing problems, Ruprecht Karls Universitaet Heidelberg, URL: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.
[27] Vasile, M., Minisci, E., Locatelli, M., 2011.: An Inflationary Differential Evolution Algorithm for Space Trajectory Optimization. IEEE Transaction on Evolutionary Computation 2(2), 267-281.
[28] Wang, H., 2007. Solving Symmetrical and DisSymmetrical TSP based on Ant Colony Algorithm. Retrieved from MATLAB® CENTRAL, URL: http://www.mathworks.com/matlabcentral/fileexchange/14543.
[29] Yang,Z.Z., Yu, B., Cheng, C.T., 2007. A Parallel Ant Colony for Bus Network Optimization. Computer-Aided Civil and Infrastructure Engineering 22, 44-45.
[30] Zhang, X., Tang, L., 2008. A New Hybrid Ant Colony Optimization Algorithm for the Traveling Salesman Problem. Advanced Intelligent Computing Theories and Applications: With Aspects of Artificial Intelligence, 148-155.
[31] Zar Chi Su Su Hlaing, May Aye Khine, 2011. Solving Traveling Salesman Problem by Using Improved Ant Colony Optimization Algorithm. International Journal of Information and Education, 1(5) , 404-409.