Controllability of C. elegans's connectome

Master's Thesis

Dániel Rácz

Applied mathematician MSc

Computing science specialization

Supervisor:
Bálint Zoltán Daróczy
Informatics Laboratory,
Institute for Computer Science and Control,
Hungarian Academy of Sciences

Inner consultant:
András Benczúr
Department of Operations Research,
Faculty of Science, Eötvös Loránd University

Eötvös Loránd University
Faculty of Science
2019

Acknowledgements

Foremost, I would like to thank my supervisor, Bálint Daróczy for the support he provided to successfully finish this thesis. His expertise and enthusiasm were an essential factor in keeping my motivation high during the last 10 months.

Many thanks to András Benczúr for undertaking the role of inner consultant.

I am highly grateful to Gábor Török for his critical attitude and the inspirational discussions we had from time to time.

Finally, my deepest thanks to Anna for the patience and encouragement she maintained during my student years. With her it all makes sense.

Contents

1. Introduction

Caenorhabditis elegans (hereinafter called C. elegans) is a one millimeter long, transparent subspecies of roundworms living mostly in soil, first observed in 1900 by Émile Maupas. Despite the insignificance this tiny creature looks to represent at first glance, it has been broadly used as a model organism since the moment biologist Sydney Brenner published his results regarding the worm's genetics in 1974. As a matter of fact, C. elegans became the first multicellular organism having its entire genetical structure described, along with its full neural map (so called connectome) known. It is out of the question that examining the C. elegans connectome from a mathematical point of view is useful, as it may help neuroscientists understand properties which are deeply hidden in the structure of its nervous system.

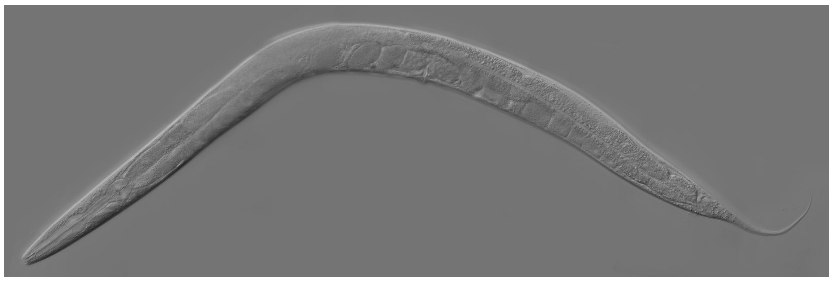


Figure 1.1: Caenorhabditis elegans

https://hu.wikipedia.org/wiki/Caenorhabditis_ elegans

The goal of this thesis is to build the sufficient theoretical knowledge for us to be able to examine the connectome of C. elegans. Concretely, we are curious about the so called controllability of the dynamical system lying behind its nervous system. Briefly, this means that we would like to know if the nervous system can reach an arbitrary state if provided the suitable shape and amount of input. In furtherance of achieving this, the formal definition of such system is given and some basic properties are introduced in chapter 2. The formal description of controllability is given as well as some fundamental theorems that are going to be the tools in our hands to describe a real-life dynamical system. One such tool is the famous theorem of Rudolf Emil Kalman from 1963, which gives an algebraic characterization of controllability ([11]). Especially the topic of structural controllability is emphasized as it really suits well such situations when the information does not hide in the precise numerical values, rather in the structure of the system. In fact, the concrete numerical properties of a real-life system can hardly be determined in most cases. The concept of structural controllability was introduced by Ching-Tai Lin in 1974 mainly based on a graph theoretic approach ([12]). Another two important milestones were reached thanks to Hosoe (1981, [10]) and Poljak (1990, [20]), whom extended Lin's theorem to a slightly more general system and gave additional equivalent forms of structural controllability. We are also having a look at the paper of Murota and Poljak (1990, [15]) in which they took the so-called target control problem into account and proved lower and upper bounds to the dimension of the largest controllable subsystem. The latter problem has been reached the attention in recent years due to its diversified applications on real-life networks. For instance, see [8] for some simple heuristics or see [9] for a more complicated heuristical algorithm on solving an even harder version of the problem and its biological application.

In chapter 3, our mathematical formalization of the connectome of C. elegans is illustrated. Some result of Albert-Lászlό Barabási and his colleagues are summarized, in which they successfully applied control theoretic results on the nerve graph of C. elegans to predict some motion related phenomena that had been unknown before. Examining the nematode's nervous system from a theoretical point of view has been occupying scientist's mind for a while now, even Paul Erdős has some papers on the topic ([17], [16], [18]). A simulation of the nervous system is implemented by the author. The code is written in Python and is available on GitHub ([1]).

Finally, in chapter 4, a fairly new approach of the problem is exposed by applying reinforcement learning on the system. Exploiting machine learning related knowledge in order to learn something about C. elegans is not an entirely new idea, for the sake of example see [7], where the authors applied a learning algorithm to approximate the dynamics of the worm's response to rapidly rising temperature.

2. Control theory

2.1 Basics

2.1. Definition. *Let $\varphi : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ be a continuously differentiable function. If the following conditions hold, then φ is called a* ***continuous time dynamical system:***

- $\forall x \in \mathbb{R}^n : \varphi(0, x) = x$
- $\forall x \in \mathbb{R}^n : \forall y, z \in \mathbb{R} : \varphi(y, \varphi(z, x)) = \varphi(y + z, x)$

Intuitively, a dynamical system is a model of a deterministic process in a way, that the state of the process after time t — starting from the state x — is given by $\varphi(t, x)$. An important variant is when the time is considered to be discrete.

2.2. Definition. *Let $\varphi : \mathbb{Z} \times \mathbb{R}^n \to \mathbb{R}^n$ be a continuous function. If the following conditions hold, then φ is called a* ***discrete time dynamical system:***

- $\forall x \in \mathbb{R}^n : \varphi(0, x) = x$
- $\forall x \in \mathbb{R}^n : \forall y, z \in \mathbb{Z} : \varphi(y, \varphi(z, x)) = \varphi(y + z, x)$

Discrete time systems can be interpreted as if the system is in the state $x(t)$ at time step t, there is an f function (depending on φ) driving the system into the state $x(t+1) = f(x(t))$. Fortunately, continuous and discrete time systems typically behave similarly.
In this thesis we are interested in the following special linear system:

$$\begin{cases} \dot{x}(t) = A(t)x(t) + B(t)u(t) \\ y(t) = C(t)x(t) \end{cases} \tag{2.1}$$

where $x \in \mathbb{R}^n, y \in \mathbb{R}^s, u \in \mathbb{R}^m, A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}$ and $C \in \mathbb{R}^{s\times n}$, respectively meaning the state vector, output and input vectors, state matrix, output and input matrices. Additionally, we also demand A, B, C and u to be piecewise continuous in t in those cases

where they actually depend on t.
The discrete time alternative is

$$\begin{cases} x(t+1) = A(t)x(t) + B(t)u(t) \\ y(t) = C(t)x(t) \end{cases} \tag{2.2}$$

having the same dimensions as above.

In chapter 2, we suppose that all the systems are linear and also time-invariant, that is if there is a Δ delay of time in the input $u(t+\Delta)$, the output will be $y(t+\Delta)$.

2.3. Notation. *For better readibility, in some cases $A(t)$, $B(t)$ and $C(t)$ are denoted by A_t, B_t and C_t, respectively.*

2.4. Theorem. (formula of variation of constants) *[6]*
Let $\dot{x}(t) = A(t)x(t) + b(t)$ be an inhomogeneous equation. The solution satisfying $x(t_0) = x_0$ is given by the following formula:

$$x(t) = \psi(t)\psi^{-1}(t_0)x_0 + \psi(t)\int_{t_0}^{t} \psi^{-1}(s)b(s)ds$$

where $\psi(t)$ is the fundamental matrix of the homogeneous equation.

Proof. Let's search the solution in the special form $x(t) = \psi(t)vt$, where $v(t)$ is an arbitrary function. Substituting this into the original equation gives

$$\dot{\psi}(t)v(t) + \psi(t)\dot{v}(t) = \dot{x}(t) = A(t)x(t) + b(t) = A(t)\psi(t)v(t) + b(t) = \dot{\psi}(t)v(t) + b(t)$$

where the last equation is a known property of the fundamental matrix. This gives us

$$\psi(t)\dot{v}(t) = b(t) \implies \dot{v}(t) = \psi^{-1}(t)b(t) \implies v(t) - v(t_0) = \int_{t_0}^{t} \psi^{-1}(s)b(s)ds$$

for some real t_0. The desired result comes after multiplying both sides by $\psi(t)$. □

2.5. Proposition. *[11] The system defined in 2.1 determines a dynamical system.*

Proof. Immediately follows from the formula of variation of constants. □

2.6. Definition. *[4] A linear system of form 2.1 is said to be* ***controllable at time*** *t_0, if there exists an input function $u(t)$ such that it transfers the initial $x(t_0)$ state into 0 in a finite time $t_1 > t_0$.*
The system is ***(completely) controllable*** *if it is controllable in all $t \in \mathbb{R}$.*
The system is ***output controllable****, if for every $y \in \mathbb{R}^m$ exists an input function $u(t)$ and a finite time $t_1 > t_0$ such that $y = y(t_1)$.*

An equivalent definition of complete controllability is given.

2.7. Proposition. *A linear system of form 2.1 is completely controllable iff for every $x \in \mathbb{R}^n$ exists a function $u(t)$ and a finite time $t_1 > t_0$ such that the corresponding solution satisfies $x(t_1) = x$, i.e. the system can be forced to an arbitrary state from its initial state.*

Proof. Let us suppose that a linear system is controllable. We have to show that we can force an initial x_0 state to an arbitrary x_1 state by finding the proper $u(t)$. Let ψ be the fundamental matrix of $\dot{x}(t) = A(t)x(t)$. Let's assume that at time t_0 we are at the state $s_0 = x_0 - \psi(t_0)\psi^{-1}(t_1)x_1$, choosing t_1 in a way that it takes t_1 time to reach 0 from s_0. Let u denote the corresponding input function. By the formula of variation of constants:

$$0 = \psi(t_1)\psi^{-1}(t_0)s_0 + \psi(t_1)\int_{t_0}^{t_1} \psi^{-1}(s)B(s)u(s)ds =$$

$$= \psi(t_1)\psi^{-1}(t_0)x_0 - x_1 + \psi(t_1)\int_{t_0}^{t_1} \psi^{-1}(s)B(s)u(s)ds$$

By rearranging x_1 to the left side one can see that u takes the
system from x_0 to x_1. □

From now until we explicitly state the opposite, we will assume that A, B, C and u are independent from t. The following theorem is a widely famous characterization of controllability.

2.8. Theorem. (Kalman's condition) *[11][5]*
Let $M = [B|AB|A^2B|...|A^{n-1}B] \in \mathbb{R}^{n\times nm}$. The system $\dot{x}(t) = Ax(t) + Bu$ is controllable iff $rank(M) = n$. M is called the controllability matrix of the system.

2.9. Notation. *The rank of the matrix M defined in 2.8 is denoted by $d(A, B)$.*

Proof of Kalman's condition theorem can be found in [11].

2.10. Corollary. Let $M = [B|AB|A^2B|...|A^{n-1}B] \in \mathbb{R}^{n\times nm}$. The system

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu \\ y(t) = Cx(t) \end{cases} \tag{2.3}$$

is output controllable iff C has full row-rank and $rank(CM) = s$. CM is called the output controllability matrix of the system.

2.11. Notation. *Let $A \in \mathbb{R}^{m\times n}$ be a matrix, then $Im(A)$ denotes the image (or range) of A, i.e. $Im(A) = \{z \in \mathbb{R}^m \,|\, \exists x \in \mathbb{R}^n : Ax = z\}$.*

Proof. Using the notations of 2.1, if the row-rank of C is smaller, than s, one can transform C into the form of having at least one row of zeros (e.g. Gauss-elimination). It is clear that along the dimension of that row there cannot be any nonzero output. Hence, C is supposed to have row-rank of s. Now using 2.7 and Kalman's criteria, we have to show that $rank(M) = n \Leftrightarrow rank(CM) = s$. But

$$rank(CM) = dim(Im(CM)) = dim\{y \in \mathbb{R}^s | \exists x \in \mathbb{R}^{nm} : CMx = y\} =$$

$$= dim\{y \in \mathbb{R}^s | \exists z \in Im(M) : Cz = y\} = s \Leftrightarrow dim(Im(M)) = n$$

because C has full row-rank. As $dim(Im(M)) = rank(M)$, we are done. $\square$

2.12. Notation. *The rank of the matrix CM defined in 2.10 is denoted by $d(A, B, C)$, while the system itself is denoted by (A, B, C).*

An important aspect of the previous proof is that a controllable system is not automatically output controllable (consider the case where does not have full rank). The converse is not true either, i.e. an output controllable system is not necessarily controllable. The reason behind it is that controllability regards to the inner state of the system, while output controllability regards to the output state. It can easily happen that a specific output state can be reached from several different inner states such that we cannot conclude these inner states only using the knowledge of our output state.

2.2 Structural controllability

2.2.1 Motivation

Following the terminology introduced in [8], a **target control** problem is a special output control problem, where one wants to control a specific set of target nodes corresponding to the rows of the matrix C (considering a system of form 2.1). Moreover, we usually want to find a minimum set of so-called driver nodes, i.e. a set of nodes described by B, for which the target node set can be controlled.
Seemingly, there is no task to perform here as only 2.10 has to be checked, however, there are two major difficulties in practical terms. One is that in case of real life networks, weights and signs of matrices A, B and C cannot be determined precisely. Another problem is that even if these matrix entries were exactly known, computing the rank of the controllability matrix may take too long in case the network is tremendously large. Finding a minimum set of driver nodes can also be quite a challenge for larger networks, as one has to examine $2^N - 1$ distinct node combinations. One way to make progress is given by some results of structural controllability.

2.13. Definition. *[12][24] Matrices M and N are said to be* ***structurally equivalent*** *if they share the same size and for every i and j $m_{ij} = 0$ iff $n_{ij} = 0$. A system (A, B, C) is structurally equivalent to a system (A', B', C') if the corresponding matrices are pairwise structurally equivalent. A linear system of form 2.1 is* ***structurally controllable*** *if the matrix $[A|B|C]$ is structurally equivalent to an $[A'|B'|C']$ matrix which defines an output controllable system. The definition is analogue for output controllability.*

The natural question arising here asks if it is worth examining structural controllability in order to tell something about the controllability of the system. The answer is yes and based on the following.

2.14. Definition. *[21] A V subset of $\mathbb{R}^n$ is called a(n) (algebraic)* ***variety,*** *if there exist some polynomials $p_1, ..., p_k$, $p_i \in \mathbb{R}[x_1, ..., x_n]$, $k < \infty$, such that*

$$V = \{x \in \mathbb{R}^n \,|\, p_i(x) = 0, \forall\, 1 \le i \le k\}.$$

That is, V is the set of common zeros of some (finite many) polynomials. V is proper if $V \ne \mathbb{R}^n$, while V is nontrivial if $V \ne \emptyset$.
A ***property*** *P is a $\mathbb{R}^n \to \{0, 1\}$ function, where $P(x)$ indicates whether the property P holds or fails at x.*

2.15. Proposition. *Let V be a proper variety. The Lebesgue measure of V is 0.*

Proof. [3] Let's suppose for contradiction that V has positive Lebesgue measure and let there be a point v in V such that v has an open, proper neighborhood - denoted by D - entirely lying in V. Let $p \in \mathbb{R}[x_1, ..., x_n]$ be one of the defining polynomials of V. By the definition of V, p is zero on D. Let $q_w(z) = p(w_1 z, ..., w_n z)$ be a polynomial with just one variable ($w \in \mathbb{R}^n$). As p is zero on D, q_w is zero on infinite many points, meaning that q_w has to be the zero polynomial. However, w can be chosen in a way that the lexicographically greatest monomial in p cannot get cancelled, leading to a contradiction. □

Consider a system $S = (A, B, C)$, where A, B and C are structured matrices, i.e. the nonzero entries are a independent parameters, while the positions of zero entries are fixed. Let these parameters be denoted by $(y_1, ..., y_a)$. Let M be the controllability matrix of S, and let $p \in \mathbb{R}[y_1, ..., y_a]$ be the sum of squares of the maximal order minors of M. Note, that this is indeed a polynomial. It is well known, that an A matrix has rank a, iff there is a non-zero minor in A of size a, while all minors having greater order are zero. Combining Kalman's condition (2.8) with this fact, we get that S is uncontrollable at $x \in \mathbb{R}^n$ (this means that we substitute x_i to y_i for all i), iff $p(x) = 0$. But this last equation defines a proper variety, hence the cardinality of those cases, where an uncontrollable system is structurally controllable has Lebesgue measure 0. Conversely, the nonzero elements of an uncontrollable, but structurally controllable system can be modified by an arbitrarily small value to obtain a controllable system. Note, that due to 2.10, the above described argument holds for structural output controllability as well. This result tells us that it is certainly useful to investigate the structural controllability of a system, especially if the exact values of the defining matrices are unknown as it commonly happens in case of real life networks. In the remaining part of this chapter, some important theoretical result is presented.

2.2.2 Lin's structural controllability theorem

The following two examples play an important role in the characterization of structural controllability given by Lin in 1974. In this section, the definitions, examples and theorems are based on [12].

2.16. Notation. *(A, b) denotes the system of the form*

$$\dot{x}(t) = Ax(t) + bu \tag{2.4}$$

where $u \in \mathbb{R}, A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$.

2.17. Example. Let (A, b) be a linear system described in 2.4. Let A and b be of the form

$$A = \begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix}$$

and

$$b = \begin{pmatrix} 0 \\ b_2 \end{pmatrix}$$

where $A_{11} \in \mathbb{R}^{k\times k}$ and $b_2 \in \mathbb{R}^{n-k}(1 \leq k \leq n)$. By Kalman's criteria (2.8), $d(A, b) < n$, hence the system is not structurally controllable.

2.18. Example. Let (A, b) be a linear system described in 2.4. Let $[A|b]$ be of the form

$$[A|b] = \begin{pmatrix} P_1 \\ P_2 \end{pmatrix}$$

where $P_2 \in \mathbb{R}^{(n-k)\times(n+1)}$ and $P_1 \in \mathbb{R}^{k\times(n+1)}(1 \leq k)$ such that P_1 can have at most $k - 1$ nonzero columns. As $rank([A|b]) < n$, the system is not structurally controllable.

Before introducing Lin's theorem, we have to get familiar with the necessary, graph theoretical terminology.

2.19. Definition. *The graph $G(A, b)$ of a system (A, b) is defined as follows. Each vertex of $G(A, b)$ correspond to a column of the matrix $N = [A|b]$, while there is a directed edge between v_i and v_j iff n_{ji} is a nonzero parameter in N. The node corresponding to the last column of N is called the origin of $G(A, b)$ and is denoted by v_{n+1}.*

2.20. Definition. *A node u in the graph G of an (A, b) linear system is called* ***inaccessible,*** *if there is no directed $v_{n+1}u$ path in G.*

2.21. Remark. Let $L = (A, b)$ be a linear system described in example 2.17, and let its graph be G. As A_{22} is the zero matrix, there is no directed edge pointing from $\{v_{k+1}, ..., v_{n+1}\}$ to $\{v_1, .., v_k\}$, therefore G contains a inaccessible vertex.
What is more, the reverse is also true, i.e. if a system S has the inaccessible vertices $v_1, ..., v_k$ in its graph, the submatrix $[A_{22}|b_1]$ must be the zero matrix (one can permutate the coordinates in a way that the inaccessible nodes have the indices $1, .., k$).

2.22. Notation. *Let $D = (V, A)$ be a digraph and $S \subseteq V$. The notation $T(S) = \{v \in V \mid \exists\, u \in S : \, vu \in A\}$ is used.*

Note, that the set $S \cap T(S)$ is not necessarily empty.

2.23. Definition. *Let G be the graph of and (A, b) linear system. We say that G contains a* ***dilation*** *if there exists an $S \subseteq V(G) \setminus \{v_{n+1}\}$ such that $|T(S)| < |S|$.*

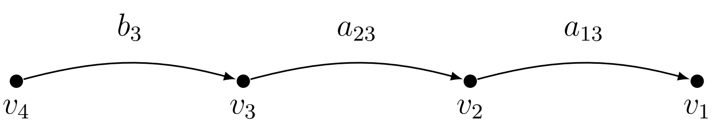
2.24. Remark. Let $L = (A, b)$ be a linear system described in example 2.18, let its graph be G, and let $S = \{v_1, ..., v_k\}$. By examining $T(S)$, one can see that G contains a dilation. Indeed, edges pointing towards S can only exist due to the nonzero entries of P_1, but P_1 can have at most $k - 1$ nonzero columns, hence $|T(S)| \leq k - 1 < k = |S|$.
The reverse is true here as well, as if a vertex set S causes the dilation in G, one can permutate the coordinates so that $S = \{v_1, ..., v_k\}$. As $|T(S)| \leq k - 1$, P_1 can have at most this many nonzero columns.

2.25. Definition. *A directed path is called a* ***stem,*** *if the starting vertex of the path is the origin of the graph.*

2.26. Example. Let the defining matrices of the system (A, b) be

$$A = \begin{pmatrix} 0 & a_{12} & 0 \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{pmatrix} \qquad b = \begin{pmatrix} 0 \\ 0 \\ b_3 \end{pmatrix}$$

The graph of the system is a stem:

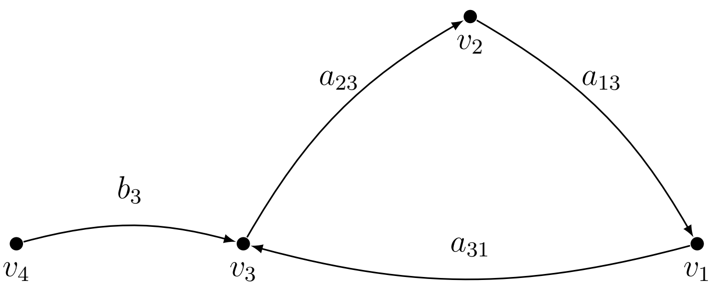


2.27. Definition. *A graph that is made of a directed cycle and a distinguished edge pointing to one of the vertices of the cycle is called a* ***bud****.*

2.28. Example. Let the defining matrices of the system (A, b) be

$$A = \begin{pmatrix} 0 & a_{12} & 0 \\ 0 & 0 & a_{23} \\ a_{31} & 0 & 0 \end{pmatrix} \quad b = \begin{pmatrix} 0 \\ 0 \\ b_3 \end{pmatrix}$$

The graph of the system is a bud:



2.29. Definition. *Let G be a directed graph having one of its vertex called the origin. Let S_0 be a stem, $B_1, ..., B_k$ are buds and u_i be the source of the distinguished edge of B_i. If for all i,*

- *u_i is not the last vertex of S_0,*
- *$B_i \cap \big(S_0 \cup \bigcup_{j=1}^{i-1} B_j\big) = u_i$,*
- *u_i is the source of an edge in $\bigcup_{j=1}^{i-1} B_j$*

then $S_0 \cup \bigcup_{j=1}^{j-1} B_j$ is called a ***cactus****. The graph G' of a linear system (A', b') is* ***spanned by a cactus****, if one can obtain a cactus by removing some edges of G'.*

We are now ready to state the following theorem.

2.30. Theorem. (Lin's structural controllability theorem) *[12]*
Let $L = (A, b)$ a linear system, the graph of L is denoted by G. The following statements are equivalent.

1. *L is structurally controllable*
2. *The coordinates of L cannot be permutated to take L into the form of either 2.17 or 2.18*

3. G does not contain any dilation or inaccessible node

4. G is spanned by a cactus

Proof. (Sketch)[12]
1. $\Rightarrow$ **2.** Proved in 2.17 and 2.18.
2. $\Leftrightarrow$ **3.** Proved in 2.21 and 2.24.
3. $\Rightarrow$ **4.** Only the main ideas of the proof is described here. Let G be a graph of a linear system, having no inaccessible nodes, nor dilation and let G be minimal in a sense that any edge deletion causes dilation or creates a inaccessible node.

2.31. Lemma. *For all $u \in V(G)$ there is exactly one directed $v_{n+1}u$ path in G (where v_{n+1} is the origin of G).*

This can be proven by supposing for contradiction that there exists a $w \in V(G)$ such that there are two different $v_{n+1}w$ paths in G. Let e be the last edge of the first path which is not an edge of the second path. By deleting this edge, G must contain a dilation (G was minimal and no inaccessible node could have been created). By examining the possible properties of this dilation, one can have a contradiction.

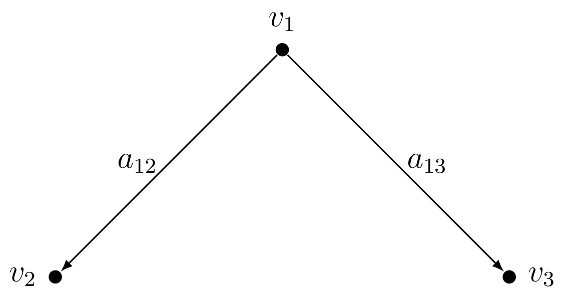
This lemma motivates the upcoming idea. Let $F = \{v_{n+1}w \mid v_{n+1}w \in E(G)\}$. For each $e_i \in F$, let $V_i = \{w \in V(G) \mid \exists v_{n+1}w \, \text{directed path in} \, G\}$. Lemma 2.31 and the fact that all nodes are accessible in G imply that $V(G) = \dot{\bigcup_i} V_i \cup \{v_{n+1}\}$. Similarly, if the subgraph of G spanned by $V_i \cup \{v_{n+1}\}$ is denoted by G_i, then $G = \dot{\bigcup_i} G_i$. These G_i graphs are called **bunches**. A G_i bunch is called **terminal bunch**, if there exists an $S \subseteq V_i$ such that $|T(S)| = |S|$. It is possible to show that there is exactly one terminal bunch in G, moreover, this terminal bunch is spanned by a cactus.

One last definition is needed. A graph $H = \dot{\bigcup_i} B_i$ is called a **precactus**, if for all i,

- B_i is a bud
- $B_i \cap \big(\bigcup_{j=2}^{i-1} B_j\big) = u_i$,
- u_i is the source of an edge in $\bigcup_{j=2}^{i-1} B_j$.

On the first hand, one can show that a precactus is spanned by a cactus. On the other hand, one can also show that all non-terminal bunches of G is spanned by a precactus. Therefore, G is spanned by a cactus.
4. $\Rightarrow$ **1.** Let G be a graph of a structurally controllable system, while B be a bud and let u denote the source of its distinguished edge. Suppose, that $G \cap B = \{u\}$. It is possible to show that in this scenario, $G \cup B$ is structurally controllable. The proof of this statement relies on the theorem saying that an (A, b) system is completely controllable, iff the implication $c^T A = zc^T \implies c^T b \neq 0$ is true, where z is a complex number and c is

a nonzero complex vector.
Considering the definition of a cactus, the above described proposition completes the proof.
For the comprehensive proof of Lin's theorem, see [12]. □
It is quite clear-cut that a network containing inaccessible nodes cannot be controlled. As for networks containing a dilation, intuitively one cannot control a set of nodes by influencing a smaller set of nodes. Let us consider the following situation.



By choosing $S = \{v_2, v_3\}$, obviously $T(S) = \{v_1\}$. While Lin's theorem tells us the system is not structurally controllable, even if we weren't aware of this theorem, we wouldn't expect the two nodes to reach an arbitrary state by controlling one node only, as we don't have effect on the distribution of the signal propagating away from v_1.

2.2.3 Controllable subspaces

At first in this section, we are examining the system of form

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.5}$$

where $u \in \mathbb{R}^s, A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$, the latter two are structured matrices. Similarly to 2.4, the notation (A, B) is used. Let n_A and n_B denote the number of parameters in A and B, respectively (recall that a structured matrix contains fixed nonzero entries and non-fixed, independent parameters). If (A, B) is not structurally controllable, one may would like to take the largest controllable subsystem into account. However, it might be unclear how to define the size of a subsystem at first glance. To resolve this problem, Hosoe introduced the concept of generic dimension in 1980 ([10]).

2.32. Definition. *Let M be a structured matrix containing m nonzero parameters. The **generic rank** of M is $genrank(M) = \max_{x\in\mathbb{R}^m} \left[rank(M(x))\right]$, where $M(x)$ denotes the result matrix of substituting the coordinates of x into the parameters of M.*

2.33. Definition. *[10] The **generic dimension** of the (largest) controllable subsystem (or subspace of the parameter space) of (A, B) is defined as $d_c = genrank(M)$, where M is the controllability matrix of (A, B).*

The motivation behind this definition is the following. Let A' and B' are concrete realizations of A and B, and let $r = rank(M) < n$, where M is the controllability matrix. let A^* and B^* be the submatrices of A and B created by r independent rows of them. According to Kalman's condition (2.8), the system defined by A^* and B^* are completely controllable. This means that in case of a concrete realization of the parameters, the dimension of the controllable subspace is given by the rank of the controllability matrix. However, this dimension is almost always equal to d_c. To see this, we can apply the same argument we used in the previous section to clarify the motivation behind the definition of structural controllability. Namely, let $P(A, B)$ be the sum of squares of the d_c-order minors of M. As P is a polynomial of the nonzero parameters, the equation $P = 0$ defines an algebraic variety denoted by V. One can see that a concrete realization of A and B having smaller dimension of its controllability subspace than d_c must lie on V.
As for the definition of the graph of a system, we can extend the definition given in the previous section (2.19).

2.34. Definition. *The graph $G(A, B)$ of a system (A, B) is defined as follows. Each vertex of $G(A, B)$ correspond to a column of the matrix $N = [A|B]$, while there is a directed edge between v_i and v_j iff n_{ji} is a nonzero parameter in N. The nodes corresponding to the columns of B are called the origins of $G(A, B)$ and are denoted by $V_B = \{v_{n+1}, ..., v_{n+m}\}$, respectively. The rest of the nodes are denoted by $V_A = \{v_1, ..., v_n\}$.*

2.35. Notation. *Let $M \in \mathbb{R}^{n\times m}$ be a matrix. The notation $M_{i_1,...,i_k}$ denotes the submatrix of M formed by rows $i_1, ..., i_k$ and columns $i_1, ..., i_k, n + 1, ..., m$ of M.*

2.36. Theorem. (Hosoe) *[10] Let $L = (A, B)$ a linear system described above and let d_c be the generic dimension of the controllable subsystem. If L cannot be permutated to the form of 2.17, then*

$$d_c = \nu([A|B]) := \max\{s \in \mathbb{Z} \mid \exists\, i_1, ..., i_s : [A|B]_{i_1,...,i_s} \textit{ has generically full rank}\}.$$

We omit the proof here as it is rather technical, nonetheless it can be found in [10]. Furthermore, we will use the following corollary only.

2.37. Corollary. The generic dimension of the controllable subsystem equals to the maximum number of accessible nodes of V_A which can be covered by a cactus configuration. A cactus configuration is a set of disjoint cycles and stems.

Proof. By the definition of ν, $[A|B]_{i_1,...,i_s}$ has generically full rank iff it contains at least $\nu([A|B])$ independent nonzero parameters. Hosoe's theorem finishes the proof as these independent entries form a cacti configuration. $\square$

The next step towards our goal to describe the system 2.1 is extending our definition of the graph of a system in order to give yet another characterization of the controllable subspace. The following construction is defined over the discrete time variant of the system 2.5, i.e.

$$x(t+1) = A_t x(t) + B_t u(t) \tag{2.6}$$

where $u \in \mathbb{R}^s, A_t \in \mathbb{R}^{n\times n}$ and $B_t \in \mathbb{R}^{n\times m}$, $1 \leq t \leq T$. From now on, we allow the defining matrices to be time dependent. The only restriction is that A_t and B_t have to be structurally equivalent to A and B, respectively.

2.38. Definition. *[20] The* ***dynamic graph*** *or* ***linking graph*** *of a system of form 2.6 - denoted by G_T^d - has the vertex set $V(G_T^d) = V_A^d \cup V_B^d$, where*
$V_A^d = \{v_i^{t,A} \,|\, 1 \leq i \leq n,\, 1 \leq t \leq T\}$ and
$V_B^d = \{v_i^{t,B} \,|\, 1 \leq i \leq m,\, 1 \leq t \leq T-1\}$,
while has the edge set $E(G_T^d) = E_A^d \cup E_B^d$, where
$E_A^d = \{v_j^{t,A} v_i^{t+1,A} \,|\, a_{ij} \neq 0,\, 1 \leq t \leq T-1\}$ and
$E_B^d = \{v_j^{t,B} v_i^{t+1,B} \,|\, b_{ij} \neq 0,\, 0 \leq t \leq T-1\}$.
A set of node disjoint paths originating from V_B^d while ending in a node $v_i^{T,A}$ for some i is called a ***linking****. The size of linking is the number of paths it contains. One such path is called a* ***legal path****.*

The idea behind this concept is that a legal path $\{v_{i_1}^{t,B}, v_{i_2}^{t+1,A}, ..., v_{i_k}^{T,A}\}$ correspond to a signal starting at time step t at an origin node v_{i_1} and arriving at node v_{i_k} under $T-t+1$ time steps. Hence, in a linking every destination of a path can be controlled independently. As a result, there is a direct connection between d_c and the maximal linking size of G_T^d, described by the following theorem.

2.39. Notation. *Let $V_{A,T}^d$ denote the set $\{v_i^{t,A} \in V_A^d \,|\, t = T\}$.*
An (S,T) linking is a linking such that the starting vertices of its paths are in S and the ending vertices are in T.

2.40. Theorem. (Poljak) *[20]*
Let L be a linear system of form 2.6. Let d_c be the generic dimension of the controllable subsystem, G_T^d be its linking graph for $T \geq n$ time steps and $G(A,B)$ be its graph (as described in definition 2.34). The following quantities are equal.

1. *The maximum number of accessible nodes of V_A in $G(A,B)$ which can be covered by a collection of disjoint cycles and stems.*
2. *d_c, i.e. the generic dimension of the controllable subsystem*
3. *$genrank([B_{T-1}|A_{T-1}B_{T-2}|...|A_{T-1}A_{T-2}...A_1B_0])$*

4. The size of a maximal linking in G_T^d

Proof.
1. $=$ **2.** See 2.37.
2. $\leq$ **3.** It is obvious from the definition of genrank.
3. $\leq$ **4.**
The following argument is a modification of the proof of Lemma 1 in [15]. Let $M = [B_{T-1}|A_{T-1}B_{T-2}|...|A_{T-1}A_{T-2}...A_1B_0]$.
Let the weight of an edge in the dynamic graph be the corresponding matrix entry and let $w(S) := \prod_{e \in S} w(e)$ if $S \subseteq E(G_T^d)$. Also we define a fixed ordering of the linking graph's nodes, denoted by $<$. In case L is a k sized linking, let the starting vertices of the paths be $s_1, ..., s_k$. The indices of the ending vertices of the paths in L define a π_L permutation (subject to the fixed $<$ ordering of the vertices). Using this observation, let $sgn(L) := sgn(\pi_L)$ be the sign of the linking L.

2.41. Proposition. *Let I and J be a subset of the rows and columns of M with the same cardinality and let $M(I, J)$ be the submatrix of M made of these rows and columns. Then*

$$det(M(I,J)) = \sum_{\substack{L\, is\, a \\ (J,I)\, linking}} sgn(L)w(L). \tag{2.7}$$

Proof. As all A_t and B_t are structured matrices, a formally nonzero $[A_{T-1}...A_{T-k}B_{T-k-1}]_{ij}$ entry of the product matrix means that there are some $k+1$ long legal paths from $v_j^{t,B}$ to $v_i^{T,A}$. Hence, the rows of M correspond to the vertices in $V_{A,T}^d$ while the columns of M correspond to the nodes in V_B^d. Let the ith row correspond to a node $u \in V_{A,T}^d$ and the jth column correspond to the node $w_t \in V_B^d$ for some t. Then

$$M_{ij} = \sum_{\substack{p\, is\, an \\ uw\, path}} w(p)$$

But this gives use the desired result, because in the expansion of $det(M(I, J))$ the intersecting paths will have opposite sign and will cancel out. $\square$
Proposition 2.41 concludes the proof as if $d_c > 0$, there exists a $d_c \times d_c$ sized submatrix of M with a nonzero determinant, but then a linking of size d_c also exists due to 2.7.
4. $\leq$ **1.**[20]
As written in Poljak's paper, the problem described in **1.** can be reformulated into a maximum weight directed cycle partition problem, which can be solved using integer programming. Let $V(G(A, B)) = V_A \cup V_B$, $|V_A| = n$, $|V_B| = m$, and if there is any inaccessible node in V_A, let's simply get rid of those. We create the graph G' by adding the edge sets $\{v_iv_j \,|\, 1 \leq i \leq n,\, n+1 \leq j \leq n+m\}$ and $\{v_iv_i \,|\, 1 \leq i \leq n+m\}$ to $G(A, B)$.

The $c(e)$ cost of the original e edges shall be 1, while the cost of the recently added edges shall be 0. One can see that a cycle cover of a vertex set U can be bijectively mapped to a cover of disjoint cycles and stems by removing the 0 cost edges. The optimal solution of the following integer program gives us a maximal cycle cover. We want to find

$$OPT_P = \max_{x \in \{0,1\}^{|E(G')|}} \sum_{e \in G'} c(e) x_e \tag{2.8}$$

where x_e is the coordinate of x corresponding to the edge e. The constrainst are

$$\forall v \in V(G') : \sum_{\substack{\exists u \in V(G') \\ e = vu}} x_e = 1 \tag{2.9}$$

$$\forall v \in V(G') : \sum_{\substack{\exists u \in V(G') \\ e = uv}} x_e = 1 \tag{2.10}$$

2.42. Proposition. *This IP problem is defined with a totally unimodular (TU) matrix.*

Proof. Let M be the following matrix. The columns of M correspond to the edges of G', while the rows of M correspond to the vertices of G'. In fact, let the rows $2i$ and $2i+1$ correspond to a node v_i such that $m_{ik} = 1$, if v_i is the target of the directed edge e_k, and 0 otherwise. Similarly, let $m_{i+1,k} = -1$, if v_i is the source of the directed edge e_k, and 0 otherwise. Then M is a TU matrix. The reason for this is that let G'' be the graph resulting in doubling the vertices of G', concretely we create $v_{i,1}$ and $v_{i,2}$ out of v_i. If $v_j v_i$ is an edge is G', let $v_{j,2} v_{i,1}$ be a directed edge in G''. Also if $v_i v_k$ is an edge in G', let $v_{i,2} v_{k,1}$ be an edge in G''. Now the vertex-edge incidence matrix of the directed graph G'' is exactly M, therefore M is TU by the well known fact that the incidence matrix of a digraph is totally unimodular.
Let b be a $2n$ dimensional vector such that $b_{2i} = 1$ and $b_{2i+1} = -1$. The IP problem $Mx = b, x \geq 0$ is defined by a TU matrix (the constraint $x \geq 0$ can be built into M without ruining the total unimodularity), and is equivalent with the above described program. Note, that 2.9 and 2.10 among with $x \geq 0$ guarantee that we can find an optimal binary x due to the total unimodularity of the defining matrix. $\square$
By multiplying the odd rows of M by -1, the system is still TU. Using the duality theroem of linear programming, 2.8 equals to

$$OPT_D = \min_{(y,z) \in \{0,1\}^{2(n+m)}} \sum_{i=1}^{n+m} (y_i + z_i) \tag{2.11}$$

subject to

$$\forall v_i v_j \in E(G') : y_i + z_j \geq c(v_i v_j) \tag{2.12}$$

We have two variables for each v_i node in the dual problem, y_i and z_i, due to the construction given in the above proof. What is more, an optimal integer valued (y, z) vector exists because c is also integer valued (this is known as the duality theorem for TU matrices). Now let us shift the coordinates of a dual vector to achieve

$$\min_{1 \leq i \leq n} y_i = -T \tag{2.13}$$

As uv_j is a 0 cost edge in G' if $n+1 \leq j \leq n+m$, $z_j \geq T$ for such j indices. The last trick is defining the following S set as

$$S = \{v_i^t \in V(G_T^d) \mid -y_i \leq t \leq z_i, 1 \leq i \leq n+m\}$$

If we show that S is a vertex cut of all the linkings in G_T^d, and $|S|$ is at most the value of **1.**, we will be done. Indeed, $|S| \leq OPT_D = OPT_P$. Let $p_k, ..., p_T$ be a path originating from a vertex corresponding to time step k in G_T^d and let t be the greatest time step such that $k \leq t \leq T$ and $t < z_t$. If $t = T$, it is obvious that $p_T \in S$, otherwise $-y_t \leq t \leq z_{t+1} - 1 \leq t < z_t$, where the first inequality is the result of combining the dual constraint and the fact that $p_t p_{t+1}$ is an edge of $G(A, B)$ (hence its cost is 1). Therefore $p_t \in S$ and this is what we needed. □

What did we achieve by the Poljak's theorem proven above? Now we are able to efficiently calculate the generic dimension of the controllable subsystem. All one has to do is finding the maximal number of vertex disjoint legal paths, but it is very well-known how to reduce it to a maximum flow problem ([24]).

Finally, we reached the point where we have enough tools to start investigating the linear system of form 2.2, i.e. solving the target control problem in general. Let's start with extending our already introduced definitions to this scenario.

2.43. Definition. *Let M be the controllability matrix of the system (A, B, C). The generic dimension of the controllable subspace is defined the same ways as before, i.e. $d_c = genrank(M)$.*

The graph theoretic terminology can also be generalized.

2.44. Definition. *[15] The graph $G(A, B, C)$ of a system (A, B, C) is defined as follows. Each vertex of $G(A, B, C)$ correspond to a column of the matrix $N = [A|B]$ or a row of C. The nodes corresponding to the columns of B are called the origins of $G(A, B, C)$ and are denoted by $V_B = \{v_{n+1}, ..., v_{n+m}\}$, respectively. The rest of the nodes corresponding to the columns of A are denoted by $V_A = \{v_1, ..., v_n\}$, while the nodes corresponding to the* *rows of C are denoted by $V_C = \{v_{n+m+1}, ..., v_{n+m+s}\}$. If $v_i, v_j \in V_A \cup V_B$, then v_iv_j is a directed edge if n_{ji} is not a nonzero entry. Additionally, if $v_k \in V_C$, v_iv_k is an edge if c_{ki} is not a nonzero entry in C.*

2.45. Definition. *[15] The **dynamic graph** or **linking graph** of a system of form 2.2 - denoted by G_T^d - has the vertex set $V(G_T^d) = V_A^d \cup V_B^d \cup V_C^d$, where*
$V_A^d = \{v_i^{t,A} \,|\, 1 \leq i \leq n,\ 1 \leq t \leq T\}$,
$V_B^d = \{v_i^{t,B} \,|\, 1 \leq i \leq m,\ 1 \leq t \leq T-1\}$ *and*
$V_C^d = \{v_i^C \,|\, 1 \leq i \leq s\}$,
while has the edge set $E(G_T^d) = E_A^d \cup E_B^d \cup E_C^d$, where
$E_A^d = \{v_j^{t,A}v_i^{t+1,A} \,|\, a_{ij} \neq 0,\ 1 \leq t \leq T-1\}$,
$E_B^d = \{v_j^{t,B}v_i^{t+1,B} \,|\, b_{ij} \neq 0,\ 0 \leq t \leq T-1\}$, *and*
$E_C^d = \{u_j^Tv_i^C \,|\, u_j^T \in V_A^d \cup V_B^d,\ c_{ij} \neq 0\}$.
*A set of node disjoint paths originating from V_B^d while ending in V_C^d is called a **linking**. The size of linking is the number of paths it contains. One such path is called a **legal path**.*

We used the same notation in case of systems of form 2.2 as in case of form 2.6, because it is clear from the context which kind of system we are talking about. After having a look at the previous definition and the theorem 2.40, a realistic expectation would be to extend that theorem for the target control problem. However, according to [24], it has not yet been comprehensively resolved as of today. The final segment of this chapter is about some partial results which can effectively be used in the next chapter.

2.46. Notation. *Following [15], the size of a maximal linking in the dynamic graph of a system (A, B, C) is denoted by $\lambda(A, B, C)$.*

2.47. Proposition. *[15] For a system (A, B, C) of form 2.2 $d_c \leq \lambda(A, B, C)$.*

Proof. The proof is almost exactly the same as the **3.** $\leq$ **4.** part of 2.40, the only difference in this case is M should be the controllability matrix of (A, B, C). □

Murota and Poljak gave an example for which $d_c < \lambda(A, B, C)$. The following lower bound is the generalization of Hosoe's theorem (2.37) for the target control situation.

2.48. Proposition. *[15] Let (A, B, C) a linear system, let $G(A, B, C)$ be its graph and let $U \subseteq V_A$ be an accessible vertex set. Let $G(A, B) \subseteq G(A, B, C)$ the graph resulting in ignoring vertices and edges that are adjacent to V_C (i.e. the graph of the subsystem of form 2.6). If U can be covered by a cactus configuration in $G(A, B)$ and U can be matched into V_C in $G(A, B, C)$, then $d_c \geq |U|$ (the latter means that a matching on $U \cup V_C$ exists such that the vertex classes are U and $W \subseteq V_C$).*

Proof. Let $M = [B|AB|...|A^{n-1}B]$. Note, that the proposition 2.41 holds for M (the proof is the same as described there). Let $e_1, ..., e_{|U|}$ be the matching between U and V_C. If we show, that the rows of M corresponding to U are linearly independent, proposition 2.41 concludes the proof as if there is an $|U|$ sized linking ending in U, each path can be extended to V_C^d by an e_i edge of the matching. But this is true, because the rows corresponding to the cactus configuration cover of V_A (and U as well) are independent due to 2.37. □

How to make use of this bound in practical situations? Besides proving it, Murota and Poljak also gave a heuristic algorithm to find a lower bound on d_c. Unfortunately, finding the maximum sized U which fulfills the requirements of 2.48 is an NP-hard problem according to [15]. However, the following theorem gives us an algorithm which is succesfully used in the next chapter.

2.49. Theorem. *[15] Let $W \subseteq V_A$ be an accessible vertex set. The problem of finding a maximal $U \subseteq W$ such that U can be covered by a cactus configuration is in* ***P***.

Proof. Let V_A' and V_B' be a copy of V_A and V_B, respectively. We are creating a H bipartite graph on the vertex set $V_A \cup V_B \cup V_A' \cup V_B'$. If w is a node in the original graph, let w' denote its copy in $V_A' \cup V_B'$. The edge set is $E(H) = E_1 \cup E_2 \cup E_3$, where

$$E_1 = \{uv' \,|\, uv \in E(G(A, B))\}$$
$$E_2 = \{uu' \,|\, u \in V_A \cup V_B\}$$
$$E_3 = \{uv' \,|\, u \in V_A,\, v \in V_B\}$$

The cost of an uv' edge shall be 1, if $uv' \in E_1$ and $v \in W$, otherwise 0. It is easy to see that a perfect matching of H bijectively correspond to a cactus configuration of a subset of $G(A, B)$, moreover, the cost of a matching equals to the number of nodes in W covered by this configuration. Hence, a maximal weight perfect matching will give us the desired U, this can be found using the Hungarian method for instance. □

Using this theorem, one may generate many maximal $W \in V_A$ subset that can be matched into V_C, then choose the one which has the greatest $U \subseteq W$ which can be covered by a cactus configuration.

2.3 Minimum set of driver nodes

In this section our goal is to find the minimum set of driver nodes necessary to make the system structurally controllable. Only the simplest case comes into consideration, particularly all state variables (equivalently all nodes of the graph) are allowed to receive an external input. Let our system be 2.5 and let $G(A)$

$$\dot{x}(t) = Ax(t) \tag{2.14}$$

where $A \in \mathbb{R}^{n\times n}$, and let its graph be $G(A)$. We would like to find out the dimension of the input vector u, which is the same as the number of columns in the matrix B, such that the system

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.15}$$

is structurally controllable. As the columns of B correspond to the driver nodes, we would like to know how many driver nodes are necessary to control the system corresponding to $G(A)$. By saying controlling $G(A)$ we formally understand controlling the system corresponding to $G(A)$. Before stating the theorem, we need the definition of directed matching.

2.50. Definition. *Let G be a digraph. We say that an $M \subseteq E(G)$ is a directed matching if it is true for pair of edges in M that neither their source, nor their target is common. A node is said to be covered by M if it is a target vertex of an edge of M. The set of nodes covered by M is denoted by V_M.*

2.51. Theorem. *[13] Let L be a linear system described above with graph $G(A)$. The minimum number of driver nodes needed to structurally control $G(A)$ equals to $max(|V(G(A))| - |V_M|, 1)$, where M is a maximum directed matching in $G(A)$.*

Proof. We would like to use Lin's theorem (2.30). If M is a perfect matching then M hast to be a directed cycle cover, because if every node is covered, then $\sum\limits_{v\in V(G)} d_{out}(v) = \sum\limits_{v\in V(G)} d_{in}(v) = |V(G)|$, hence for every $u \in V(G)$ $d_in(u) = d_out(u) = 1$. In this case, one new v_{n+1} node is added to the graph as the origin node and let one directed edge point from v_{n+1} to one vertex of each cycle. If an arbitrary edge of an arbitrary cycle is deleted, the resulting graph forms a cactus and hence is structurally controllable.
If M is not perfect, then the edges of M is a disjoint union of directed cycles and paths. Let one connect an origin node for each unmatched node, forming $N - |V_M|$ stems. This way, only the cycles has to be dealt with, but we can connect them to an arbitrary origin vertex, resulting in a disjoint cactus cover of $G(A)$. Note, that if a node set is covered by disjoint cactuses, it is covered by a directed matching determined by the cacti. Therefore, a maximum matching indeed determines the minimum set of driver nodes. $\square$

2.52. Remark. It is easy to find a maximum directed matching in a digraph G. For example, one can create make a $V'(G)$ and a $V''(G)$ copy of $V(G)$ and form a bipartite graph H on $V'(G) \cup V''(G)$. If uv is a directed edge in G, then $u'v''$ shall be an undirected edge in H, where $u' \in V'(G)$ corresponds to u and $v'' \in V''(G)$ corresponds to v. The maximum directed matchings in G naturally correspond to the maximum matchings in H that can be found using König's augmenting path algorithm of the Hopcroft-Karp algorithm (the latter is more efficient).

2.53. Remark. A more complex problem would be the following. Let an $F \subseteq \{1, .., n\}$ is also given representing the so-called forbidden variables. These variables (or equivalently these nodes in $G(A)$ cannot be directly affected by an external input signal. The question is what is the minimum set of driver nodes necessary to make the system 2.15 structurally controllable such that there is no direct edge from any driver node to any forbidden one? The answer can be found in [19] along with a polynomial time algorithm finding the minimal set of suitable drived nodes.

3. Connectome of C. elegans

In [24], the authors successfully applied control theoretic tools to a model of C. elegans's connectome. In the following sections, we will mostly be following their notations.

3.1 The model

The model described in this section is the one presented in [24]. We imagine the connectome as a directed graph with its nodes corresponding to the worm's neurons, and its edges corresponding to the physical and chemical connections of these neurons [2]. As we are interested in the moving mechanism of C. elegans, only the 282 non-pharyngeal out of all the 302 neurons together with 97 muscles are present in the model. As the authors highlight in their paper, neurons CANL/CANR and VC06 are excluded as they are not accessible from the sensory neurons according to this dataset. Also the muscles representing the anus and vulva are not considered to take part in the worm's motion mechanism.

Neurons are assigned three types, namely sensory neurons, interneurons and motorneurons. Sensory neurons are directly connected to the sensory features of the worm. In our model, we make the assumtpion that a sensory neuron make a single connection to a sensory organ. Motor neurons are the ones connected to the muscles or other organs. As we would like to study the motion mechanism of C. elegans, only the muscles are relevant for us. Interneurons are in between transmitting the signal between sensory neurons and motorneurons. These definitions might not be totally precise in a biological point of view, however are enough for a simple model of the nervous system.

The edges represent a connection between two neurons or a neuron and a muscle. These connections are directed, meaning that if uv is an arc in the graph, the corresponding neuron to u is able to send a signal to the corresponding neuron/muscle to v. Note, that in many cases uv and vu are both edges of the graph, in some rare cases even self loops are present. In reality, these connections have several different kinds, such as chemical synapses, gap junctions, neuromuscular junctions, etc. In section 3.3 some details are considered, however, in this model only the presence of a connection counts.

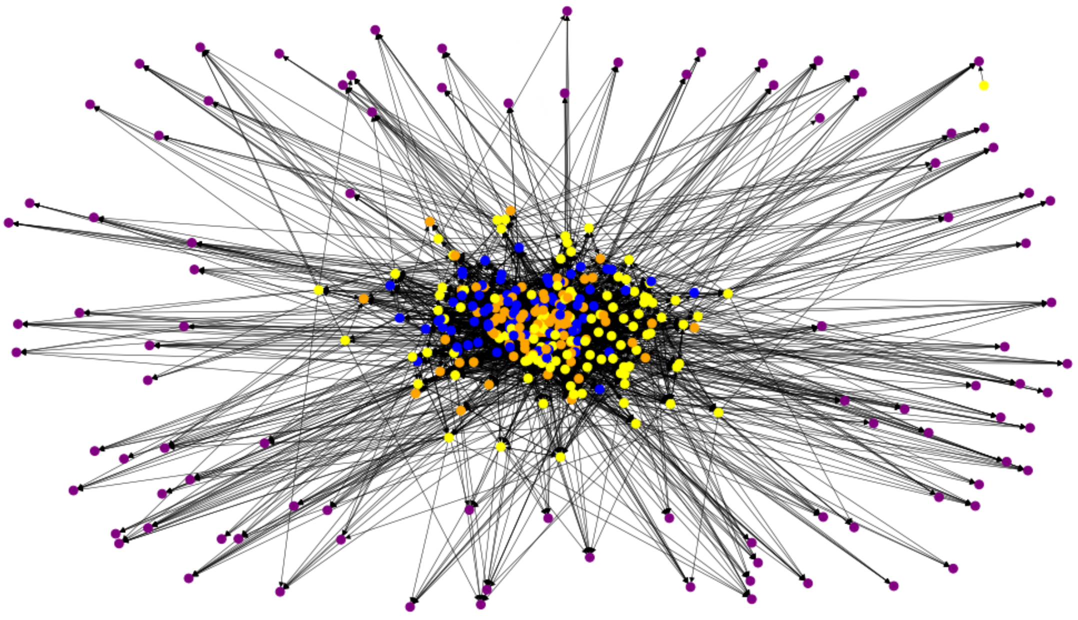


Figure 3.1: Directed graph of the nervous system

In the figure above, blue nodes represent the sensory neurons, interneurons are orange, motorneurons are yellow and muscles are purple. The positions of the neurons and muscles along the body of the worm is illustrated in the following figure.

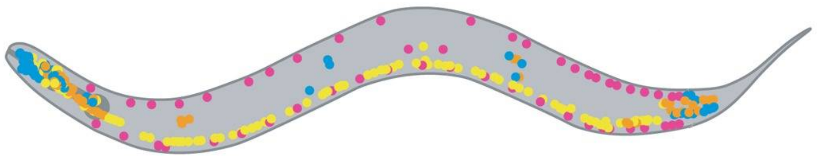


Figure 3.2: Source: [24]

For the sake of moment, let us denote the number of neurons by N and the number of muscles by M. The propagation of a signal captured by the sensory organs through the graph can be modeled by a dynamical system

$$\dot{z}(t) = f(z, v, t) \tag{3.1}$$

where $z \in \mathbb{R}^{N+M}$ is the state vector, $v \in \mathbb{R}^s$ is the input vector and f is some continuously differentiable function. The accepted standpoint is that f is a nonlinear function. In contempt of this, a linear time-invariant system is assumed to model the nervous system, viz. a system of form 2.1. In this scenario, matrices A, B and C are time independent and respectively representing the wiring of the nervous system, the sensory neurons and the muscles.

The exact values of these matrices are unknown. For example, the rows and columns of A correspond to the vertices of the nerve graph (which correspond to the neurons) such that $a_{ij} > 0$ if there is a directed edge in the graph between nodes v_j and v_i. Here, the value of a_{ij} somehow measures the efficiency of the signal transmition between neurons corresponding to v_j and v_i. Of course there is no one who exactly knows these values or even has the power to explicitly measure these. Similarly, the rows of B represents the $N + M$ nodes, while the columns represent the external signals. In this context, each sensory neuron gets (a possibly zero) signal at time t, and the elements of B are such that the ith coordinate of $Bu(t)$ represents the stimuli that v_i recieved. It is important to note, that in [24], the measure of a signal is not considered, meaning that a sensory neuron either receives a signal or not. The rows of matrix C correspond to the muscles implying that the ith coordinate of $Cx(t)$ is the state of the ith muscle at time t. Neither the exact entries of B nor C are known for the same reason as discussed before. This motivates us to handle these matrices as structured matrices and investigating the structured controllability of the system.

Before doing so, some words about why it is worth examining the linearized dynamical system at all. The informal answer to this question could be because we have more knowledege of linear systems, while knowing the linear charactheristics of the system may get us into a better position to solve the nonlinear case. This is especially true for complex real-world networks such as the human brain, where it is already a difficult problem to tell something about the linear dynamics of the system. The formal answer to the question starts at Lyapunov's classical stability theory and goes as far as modern stability theory. Briefly talking, it can be formally proven that in many cases knowing the linear stability or structured controllability of the system is sufficent to determine the nonlinear charactheristics. For the details see e.g. [22].

3.2 Applying control theory

The structure of C. elegans's nervous system has been mapped for a while now, suitably there has been much research going on this topic by many kinds of experts, but mainly neuroscientists. Accordingly, the functions of neurons or neuron classes are more or less discovered, yet we are far from the perfect knowledge of its working mechanism. One might think that the only way to get closer to this goal is by performing direct experiments with living worms under microscope. However, theoretical research also has its potential to eventuate important results. We are about to witness an example of this scenario, as a team of researchers has managed to predict that certain neurons must take part in the worm's response mechanism to anterior gentle touch ([24]). The essence of this result is that many other neurons were known to take part in this mechanism as the outcome of several experiments performed by neuroscientist, but not the newly discovered one. What is more, physical experiments were only used to confirm the results achieved by applying graph theoretic and control theoretic tools to the mathematical model associated with the nervous system of C.elegans.

According to [24], the sensory neurons responsible for detecting anterior gentle touch are *AVM, ALML* and *ALMR*. Applying 2.47 to the connectome assuming only these three neurons are allowed to receive external input, we have the linking size as an upper bound of 89 to the number of independently controllable body wall muscles. This result has also been verified by the author using the dataset [2], see the python script **upper_bound.py** in our repository [1].

As a lower bound, authors of [24] used a very similar method to the heuristic algorithm given by Murota and Poljak based on 2.48. Following their notation let the connectome graph be G, let $V_M \subseteq V(G)$ be the vertices corresponding to the muscles, $V_S \subseteq V(G)$ be the vertices corresponding to the sensory neurons, $V_D \subseteq V(G)$ be the vertices adjacent to V_M and let $V_O = V(G) \setminus V_M$. First, search for a H subgraph on $V_O \setminus V_D$ that is structurally controllable. This can be done by using the auxiliary graph described in the proof of 2.49. Next, extend H to a H' graph by adding vertices from V_D such that the extended graph is still structurally controllable. Then a lower bound is achieved by the cardinality of the maximum $U \subseteq V_D \cap V(H')$ that can be matched into V_M (see 2.48). Interestingly, a lower bound of 89 can be achieved by this heuristic method, resulting in the fact that the number of independently controllable muscles controlled by *AVM, ALML* and *ALMR* is 89 in an adult, healthy nematode. This result is also verified by the author, see the python script **lower_bound.py** in the repository [1]. A surprising subsidiary result of the above described method is that the motor neurons adjacent to the muscles are structurally (state-) controllable, meaning that they independently receive signals from the sensors. The general conjecture regarding the control of biological systems is that the target set should not be fully controllable as it is not practical for a living system. For instance, if all the worm's muscles were able to strongly contract at the same time, besides that the worm wouldn't change its place, it would probably cause some damage in the long run. The previously described result then means that the conditioning of the movement happens between the muscles and its neighbours, rather than using the state of the remaining nerve cells.

One of the main consequence of the control analysis is that the neuron *PDB* plays an important role in the response mechanism to anterior gentle touch. By important role we mean that the ablation of this neuron negatively affects the response or motion mechanism of C. elegans.

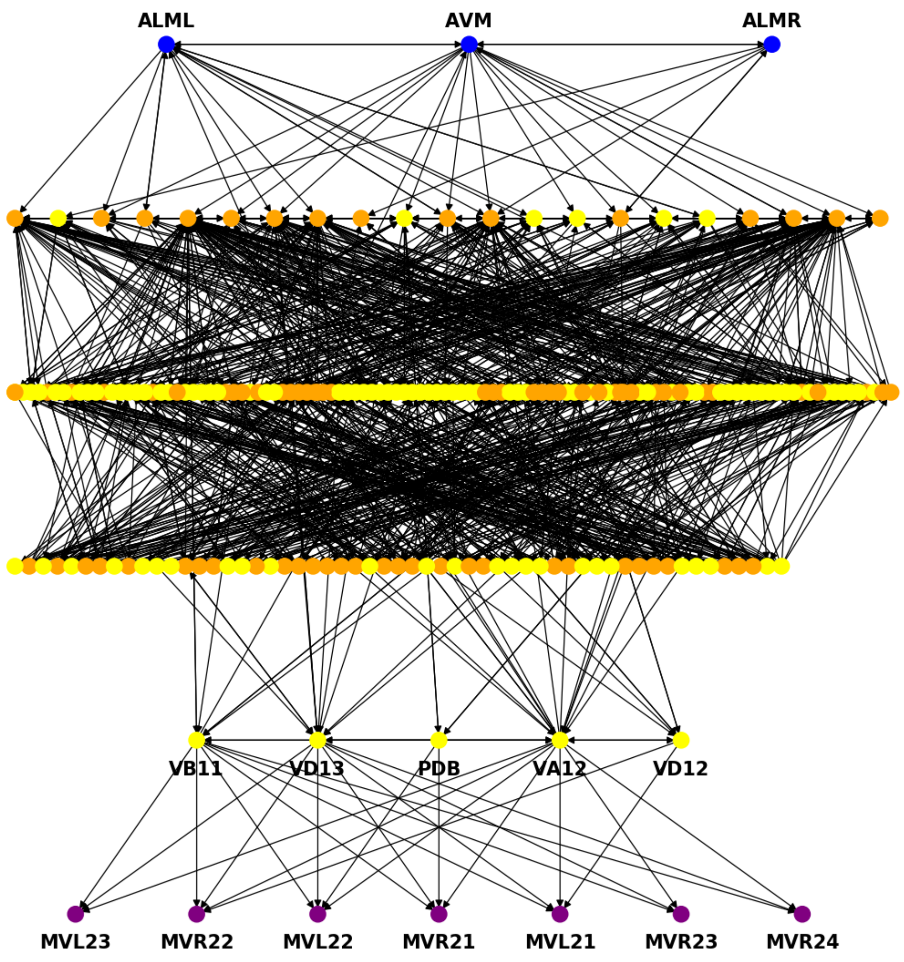


Figure 3.3: Based on a figure of [24]

In the figure above, one can see that the seven muscles are adjacent to five motor neurons only, meaning that any signal received by the sensor neurons can get to the muscles via those five motor neurons. Applying Lin's theorem (2.30), the above described network cannot be structurally controllable (not even in the complete sense). As only 5 nodes can be matched to the set of those 7 muscles, those 7 muscles cannot be independently controlled by those 3 receptor neurons. This conjecture has been confirmed by experiments performed on *PDB* ablated worms (see [24] for the corresponding video records).

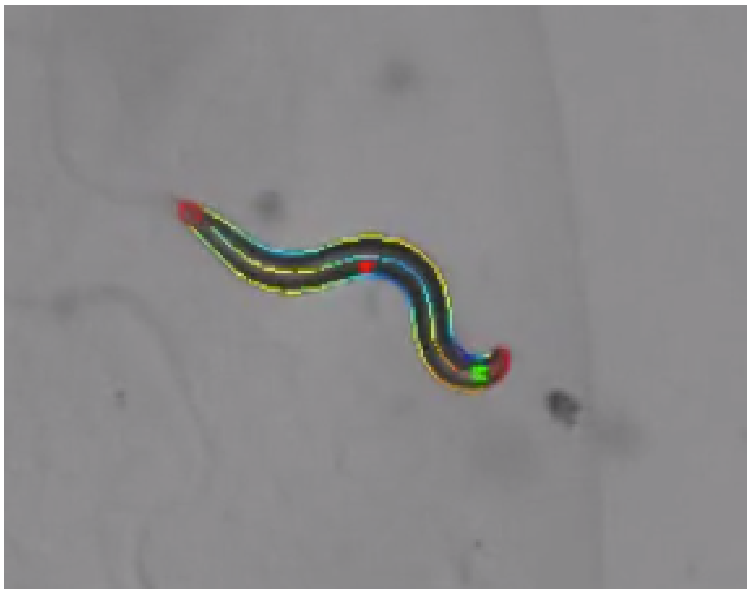


Figure 3.4: Frame from a video of a PDB ablated worm

Source: [24]

Another result presented in [24] is that some particular neurons of a neuron class called *DD* play an important role in the response mechanism to either anterior or posterior gentle touch, while some other neurons in the *DD* class can be ablated without disturbing this response mechanism. This is quite unexpected considering that the general pattern of connectivity was believed to be similar in the *DD* class.

3.3 Simulation

As there are several existing implementation of the C. elegans nerve system, it could seem strange to create another for us instead of using one of those. The main reason for this is there are many uncertainties regarding the C. elegan's nervous system behavior. Even the neuroscience experts are debating, guessing or making assumptions in many important aspects needed for a correct implementation. Thus, it was inevitable to create our own to be able to freely experiment with the parameters of the model.

The starting point for our simulation was the model introduced in section 3.1. As compared to that one, some more features are present in this simulation trying to make it more alike the real one. However, in several cases we had to apply assumptions or approximations as many details are yet unknown even for neuroscientists. In our model, we also omitted the neurons CANL/R and VC06, along with the neuron PVDR as this one is also inaccessible from the sensory neurons according to our dataset. Another important difference compared to the previous model is that we assume that the power of external stimuli can be measured, i.e. there is a numerical value of each input.

Different types of connections between neurons are considered as well, altogether we deal with three of them, namely electric junction, neuromuscular junction and chemical synapse. After looking into several resources, we decided to set the speed of the chemical synapse twice as slow as the other two. The dataset we used also contains information about the physical density of these connections, namely there is a number attribute for each connection. To see what this means, let's have a look at the very brief shape of a nerve cell.

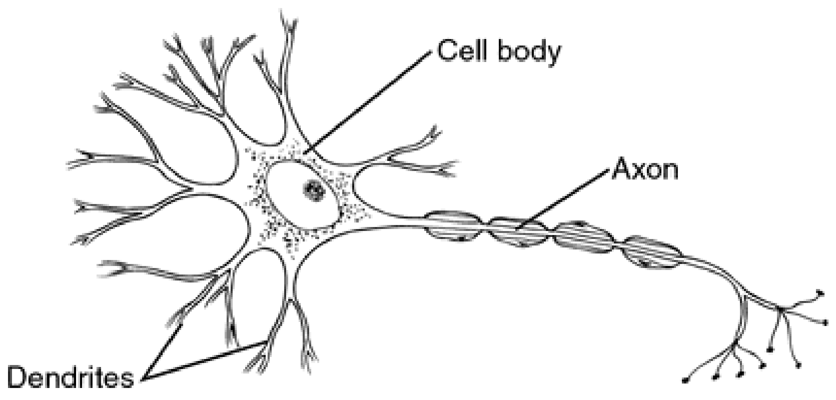


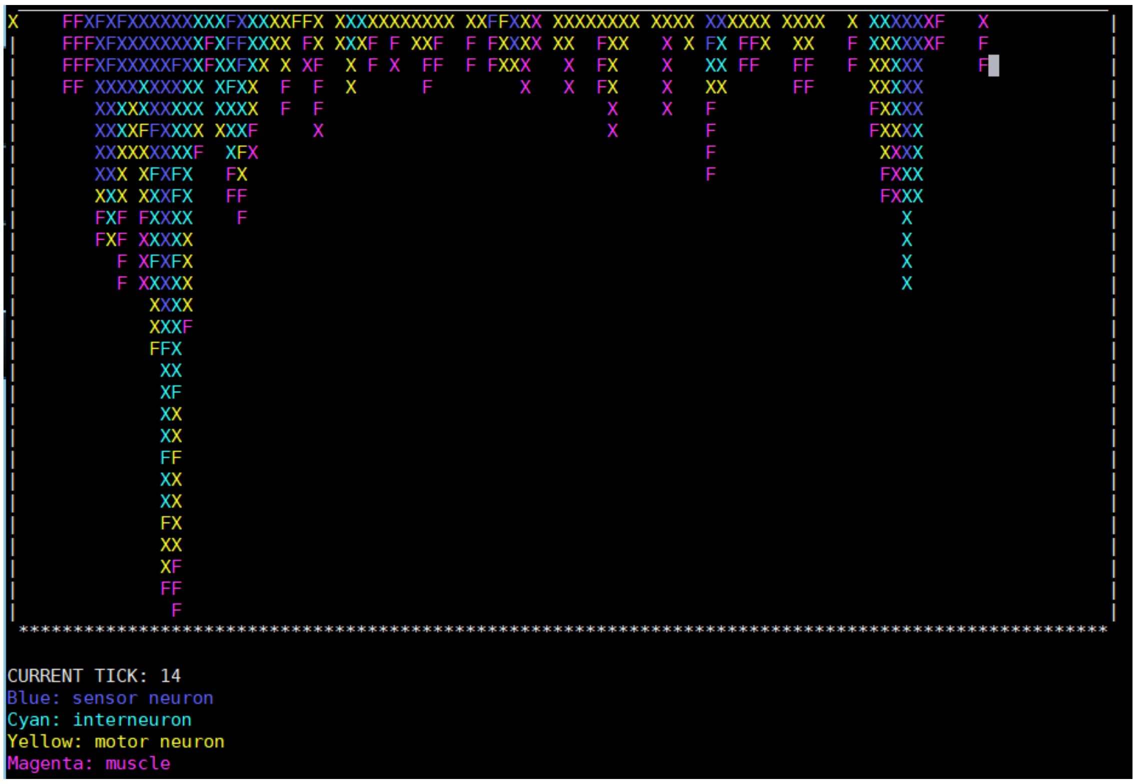
Figure 3.5: Source: https://img.tfd.com/mk/D/X2604-D-16.png

The long fiber which conducts an electrical signal away from the cell is called the axon. As an axon is able to form several branches, it is able to transmit the signal via many synaptic terminals. Similarly, the dendrite, which is the part of the cell responsible for propagating the incoming signal are split into branches. This means that the endings of the axon of a neuron can connect to a dendrite of another neuron in several different points. We built into our model the information about these connection points from the dataset.

Another important aspect is that a threshold value is given for each neuron. This means that a neuron won't forward the incoming signal, that is to say it won't fire, if the incoming signal is not strong enough. We imagine that the polarization of a neuron happens progressively, namely the neuron gathers the signals for a certain amount of time and if it reaches its threshold value, it gets depolarized and sends the amount of signal it stored to all of its neighbors. As for the muscle cells, they also have a threshold value, above which the muscle contracts. A fatigue rate is also set for the muscles, that is the value stored in a muscle decreases over time. We also made experiments with a contraction limit, i.e. if a muscle is contracted for too long, it gets tired and stretches for a while, despite the incoming signals to contract again, but this has not got into the model yet.

Propagation of a signal through the network works as follows. The time is discrete, we call one time step a tick. In the initial tick, some input value is conducted to some sensory neurons. Then, the value in a sensory neuron s is getting split into $d_{out}(s)$ part uniformly, where $d_{out}(s)$ means the out degree of the node corresponding to s in the graph of the nervous system. For each edge we have the speed of the signal being transmitted corresponding to the type of the connection set. According to this speed property, the signal on an edge gets into its destination neuron in the $(t + k)$th tick, where t is the current tick and k is the speed property of the edge. In general at time tick t, the incoming signals are summed for each neuron, then those neurons whom are above their threshold fire and forward the amount of signal they store. Firing happens instantly as we imagine neurons work much faster compared to the signal transmission speed among nerve cells. The simulation is finished, if we reach a tick limit T, or if the energy of the system has become zero. This can happen, because the propagating signal decreases over time in the muscles. Some very slow fatigue rate shall be set for the neurons as well.

One more feature of our model is that edge thresholds are also in the picture. We imagine that during its learning phase, a neuron of a real worm learns to enhance/weaken a signal arriving from a specific neighbor, or even only on a specific branch of its dendrite. This way a worm is able to fine tune its movement for example or its response to a gentle touch or a strong hit. As the information about the density of a connection is built in the model, we can make use of that. For example, let's suppose that there is a triple connection between a pair of neurons c_1 and c_2 such that c_1 sends the signal to c_2. We are able to set the threshold values individually on the 3 branches they communicate through. What is more, also the distribution of the outcoming signal among those three branches can be set. More information of how to make more use of this is in chapter 4.



```
X    FFXFXFXXXXXXXXXFXXXXXFFX XXXXXXXXXXX XXFFXXX XXXXXXXX XXXX XXXXXX XXXX  X XXXXXXF    X          |
|    FFFXFXXXXXXXXFXFFXXXX FX XXXF F XXF  F FXXXX XX  FXX   X X FX FFX  XX   F XXXXXXF    F          |
|    FFFXFXXXXXFXXFXXFXX X XF  X F X  FF  F FXXX   X  FX    X   XX FF   FF   F XXXXX      F          |
|    FF XXXXXXXXXX XFXX  F  F  X      F        X   X  FX    X   XX      FF     XXXXX                 |
|       XXXXXXXXXX XXXX  F  F                          X    X   F              FXXXX                 |
|       XXXXFFXXXX XXXF     X                          X        F              FXXXX                 |
|       XXXXXXXXXF  XFX                                         F               XXXX                 |
|       XXX XFXFX   FX                                          F               FXXX                 |
|       XXX XXXFX   FF                                          F               FXXX                 |
|       FXF FXXXX    F                                                            X                  |
|       FXF XXXXX                                                                 X                  |
|         F XFXFX                                                                 X                  |
|         F XXXXX                                                                 X                  |
|            XXXX                                                                                    |
|            XXXF                                                                                    |
|            FFX                                                                                     |
|             XX                                                                                     |
|             XF                                                                                     |
|             XX                                                                                     |
|             XX                                                                                     |
|             FF                                                                                     |
|             XX                                                                                     |
|             XX                                                                                     |
|             FX                                                                                     |
|             XX                                                                                     |
|             XF                                                                                     |
|             FF                                                                                     |
|              F                                                                                     |
 ****************************************************************************************************

CURRENT TICK: 14
Blue: sensor neuron
Cyan: interneuron
Yellow: motor neuron
Magenta: muscle
```

Figure 3.6: Simulation in progress

In the figure above, a simple graphical interface of our simulation can be seen. The left side of the rectangle correspond to the head of C. elegans, while the right side correspond to its tail. The colorful letters (X and F) are the neurons and muscles along the worm's body. F means the neuron is fired in the actual tick while X means it is not. In this scenario, all the sensory neurons received an input signal. The simulation ended at tick 24 as the system has ran out of energy.

An important use case of our simulation is it makes investigating the effect of each sensor neuron on each muscle cells possible. For instance, in the following figure the effect of *AVM* on some of the muscles is plotted. In this scenario, *AVM* received an input of 1500 unit, while the threshold values of all neurons were set to some constant value. For a muscle we can track that out of the 1500 unit, how many got to the muscle eventually. On the plot below, three types of average value are present, namely the average value during those time steps where the muscle was triggered, the average value during those time steps where any signal reached the muscle and finally, the average during the whole simulation. The figure here contains those muscles only which were triggered during the simulation.

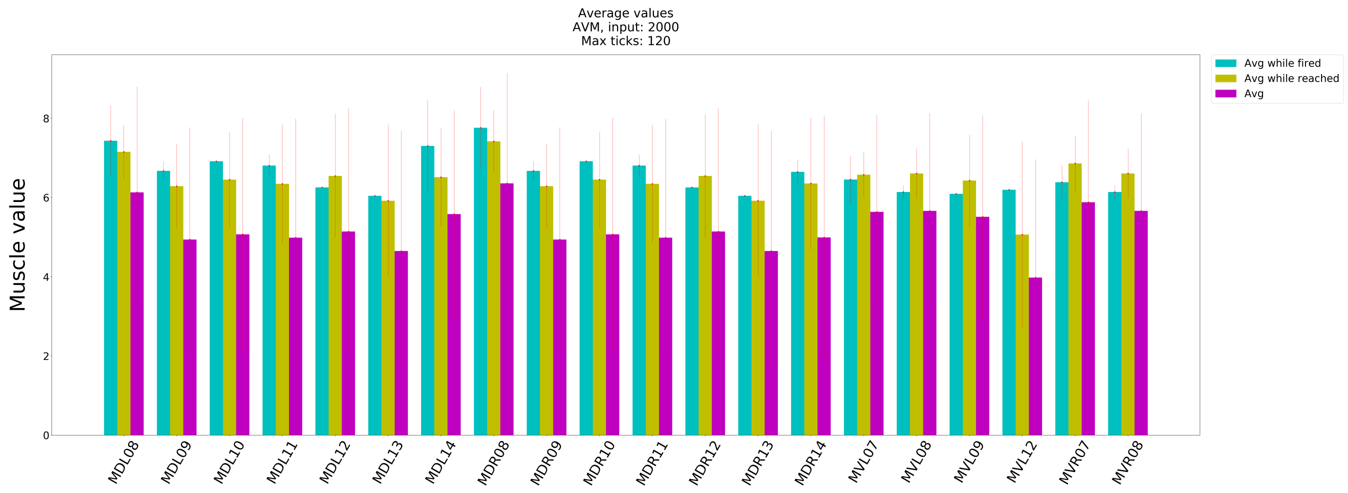


Figure 3.7: Effect of AVM on some muscles

By examining the graph containing all the muscles, not just these ones, on could see that a couple of muscles gets a relatively large portion of the signal sensed by *AVM*, while other muscles are not triggered at all. Of course, if we set the input large enough all muscles might get fired, however, we consider this an unrealistic scenario.

As for the technical details, the simulation is implemented in Python 3 and is based on the dataset [2]. The properties of the neurons and the connections between them are stored in the corresponding neuron and synapse objects. These are lumped together by a Network class containing the graph structure stored in a NetworkX instance. A simulator class is dedicated to perform the simulation itself, having the network as one of its attributes. The graphical interface is implemented using the curses package. The source code is available at [1].

4. Reinforcement learning

In this chapter, we are introducing the basic terminology of the simplest form of reinforcement learning models and the result of our experiments of applying Q-learning to our model of C.elegans's nervous system based on the simulation described in section 3.3. In the whole chapter, especially in section 4.1, we are following the notation of [23]. All theoretical result introduced here is their work and cited from their book.

4.1 A brief overview

Machine learning is the field of studying learning algorithms. The most widely accepted definition of learning algorithm was given by Tom M. Michell in 1997: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." Informally talking, tasks which are usually solved by machine learning tools cannot be solved in a classical algorithmical way, as only a concrete set of rules or logical patterns can be implemented. Some popular examples are regression or classification problems. As for the experience, it usually means the information extracted from one or more data points. The measure of performance depends on the context but usually happens using some test dataset, i.e. via comparing the output of the algorithm with the expected output.

Examining the nature of machine learning algorithms, three different class can be created. The first two would be supervised and unsupervised learning. Without going into the details, in case of supervised learning, the training dataset — the data used to train the model — is labeled, meaning the model receives the expected output for each piece of data and is able to use that during the training process. Unsupervised learning is about learning some nontrivial properties hidden in a dataset which describes the structure of the data.

The third one is called reinforcement learning and its simplest form is usually modeled as follows. There is a so-called learning agent in some kind of state, that is able to interact with is environment in order to reach a goal. Such interaction includes an action the agent performs, resulting in a possible change of its state as well as a reward it receives for its action. The reward represents the quality of the performed action. The overall goal of the agent is to maximize the cumulative reward it receives during its running.

Let us denote the state of the agent at time t by S_t, the action it performs by A_t and the reward it gets after this action by R_t. The first problem is how should an agent decide, which action to take in a given state? First, the agent has a $Q_t(a)$ estimation of the value of an action a. The real value of a is given by

$$q_*(a) = \mathbb{E}(R_t \,|\, A_t = a) \tag{4.1}$$

where the expected value is taken over the probability distribution from which the reward is chosen. Note, that this distribution is not necessarily unchanged over time. The agent has the choice to choose between the action with the highest estimated reward — which is called a greedy action — or an action with a smaller reward which may result in improving the value of this action. The latter is called exploring while the former is called exploiting. In order to maximize the reward in the long run, the agent has to find a good balance between exploiting its current knowledge and exploring other possibilities.

Formally, the running process of a reinforcement learning algorithm is interpreted as a Markov Decision Process.

4.1. Definition. *A finite **Markov Decision Process** or **MDP** is a tuple (S, A, P, R), where*

- *S is the set of states satisfying the Markov property*
- *A is the set of actions*
- *P is the set of sate-transition probabilities, i.e.*
  $P = \{Pr(S_t = s' \,|\, S_{t-1} = s, A_{t-1} = a) \,|\, s, s' \in S, a \in A, 1 \leq t \leq T\}$, where T is the final time step
- *R is the set rewards, i.e. $R = \{\mathbb{E}(R_t \,|\, S_{t-1} = s, A_{t-1} = a) \,|\, s \in S, a \in A, 1 \leq t \leq T\}$, where T is the final time step*

An $S_t = s$ state has the Markov property, if $Pr(S_{t+1} = s' \,|\, S_t = s) = Pr(S_{t+1} = s' \,|\, S_1 = s_1, ..., S_{t-1} = s_{t-1}, S_t = s)$, meaning that s has all relevant information of the past.

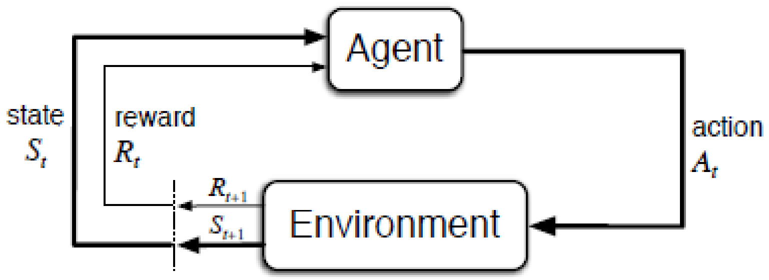


Figure 4.1: Source: [23]

There are two main approach to set the objective function the agent wishes to maximize. If the agent-environment interaction is made of subsequences, e.g. games of chess, we call these subsequences episodes. The states episodes end are called terminal states. If T is a random variable representing the terminal time step, the agent wants to maximize

$$G_t = R_{t+1} + ... + R_T \tag{4.2}$$

In this case, the task is called episodic. The opposite of episodic tasks are called continuing tasks, in which case $T = \infty$, i.e. one cannot break the interaction into subsequences. The key idea here is that the agent should discount the current value of future rewards, namely let $0 \leq \gamma \leq 1$ be the discount rate. Then

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{4.3}$$

is maximized over time. Note, that this series converges if the rewards are bounded and $\gamma < 1$.

Another important characteristic of an RL model is called the policy which determines how the agent behaves in each state. Concretely, it the policy the agent follows at time t is denoted by π_t, then $\pi_t(a, s) = Pr(A_t = a|S_t = s)$ denotes the probability of action a will be taken at state s at time step t. If t is omitted from π_t, it means that the policy is time independent (only depends on the state). During a policy learning process, the policy improves over time caused by the effect of the experiences. The value function is the estimator function the agent uses to measure how good a state or an action is under the policy it follows. Namely,

$$v_\pi(s) = \mathbb{E}_\pi(G_t \,|\, S_t = s) \tag{4.4}$$

is called the state-value function for policy π and gives the expected reward of starting in s and following π. Here the expected value is taken over the random variable representing the chance of the agent following the policy π. The action-value function can be defined in a similar manner:

$$q_\pi(s,a) = \mathbb{E}_\pi(G_t \mid S_t = s, A_t = a) \tag{4.5}$$

We say that a π_1 policy is better than or equal to a policy π_2 (denoted by $\pi_1 \geq \pi_2$), if $v_{\pi_1}(s) \geq v_{\pi_2}(s)$ holds for every $s \in S$. If π_* is better than or equal to all other policies, then π_* is an optimal policy. The corresponding optimal state-value and action-value functions are defined as

$$v_*(s) = \max_\pi v_\pi(s) \tag{4.6}$$

and

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{4.7}$$

4.1.1 Temporal-Difference Learning

Temporal-difference (or TD) learning is a fundamental method of the reinforcement learning introduced by Richard S. Sutton in 1988. We are going to use it as an implementation of the so-called generalized policy iteration method. The latter means the following. Imagine we have an algorithm denoted by E which takes a policy π as an input and outputs v_π or at least a suitable approximation of it. Computing the state-value function of a policy is called the policy evaluation or the prediction problem. Also imagine we have another algorithm I taking a policy π_0 and its state-value function as an input and outputs a π_1 policy which is strictly better, than π. The idea of policy iteration method is repeating I and E to achieve better and better policies until we reach the optimum:

$$\pi_o \rightarrow v_{\pi_0} \rightarrow \pi_1 \rightarrow \ldots \rightarrow \pi_* \rightarrow v_{\pi_*}$$

There are several ways to solve the problem of policy evaluation and policy improvement, such as dynamic programming, Monte Carlo methods or TD learning. The key thought of the latter is as follows. Let V be an approximation of v_π for some policy π, let $0 \leq \gamma \leq 1$ be a discount rate and $0 < \alpha \leq 1$ be a parameter called the learning rate. During the running process, the algorithm will make the following update

$$V(S_{t+1}) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)) \tag{4.8}$$

The origin of the name temporal-difference comes from this update method, as the estimated value of S_t depends on the discounted difference of $V(S_{t+1})$ and $V(S_t)$. It is also implicitly follows that the algorithm has to wait one time step to know $V(S_t)$. The term $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is denoted by δ_t and is called the TD error. The following pseudocode describes the so-called $TD(0)$ algorithm for policy evaluation. The notation refers to the $TD(\lambda)$ algorithm of which $TD(0)$ is the simplest special case. It is possible to show that under certain conditions, the $TD(0)$ algorithm converges to a single answer.

```
$TD(0)$ for approximating $v_\pi$
Input: $\pi, \alpha, \gamma$
Output: $v_\pi$
 1: $V(terminal) := 0; V(s)$ initialized arbitrarily for nonterminal states
 2: for all episode $e$ do
 3:     initialize $S$
 4:     for all step of $e$ do
 5:         $A \leftarrow$ action given by $\pi$
 6:         perform $A \rightarrow R, S'$ reward and state
 7:         $V(S) \leftarrow V(S) + \alpha(R + \gamma V(S') - V(S))$
 8:         $S \leftarrow S'$
 9:     end for
10: end for
```

For the policy improvement, we will describe the so-called Q-learning algorithm introduced by Christopher J. C. H. Watkins in 1989. Our task is to approximate the optimal action-value function q_*. Let Q denote the estimation function. The most important aspect of Q-learning is that it directly approximates q_* independently of the policy being followed by the agent. Of course, the policy has the effect on choosing the state-action pairs to perform, however, it is not necessary to be known in order to prove the convergence of this method.

```
$Q$-learning for approximating $\pi_*$
Input: $0 < \alpha \leq 1, \epsilon > 0$
Output: $\pi \approx \pi_*$
 1: $Q(terminal, a) := 0 \, \forall a \in A$; $Q(s, a)$ initialized arbitrarily for nonterminal states
 2: for all episode $e$ do
 3:     initialize $S$
 4:     for all step of $e$ do
 5:         $A \leftarrow$ action given by $Q$ and $S$
 6:         perform $A \rightarrow R, S'$ reward and state
 7:         $Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma \max_a Q(S', a) - Q(S, A))$
 8:         $S \leftarrow S'$
 9:     end for
10: end for
```

Under certain conditions, Q-learning provides a convergence with probability 1.

4.2 Experiments

By using the simulation introduced in section 3.3 and the algorithm of Q-learning introduced in section 4.1, it is possible to try to teach the nerve graph some features in contemplation of understanding how the real worm's nervous system evolves from the point it is born. The larges obstacle in our way to do so is that this would require a very precise tuning of the hyperparameters of our model. For instance, we do not know the exact threshold value of the neuron, nor the threshold of muscles. We suspect that it would be enough to know these values only compared to each other and the input sizes, however, we have no information regarding these values. These observations do not really allow us to call our graph model the model of C. elegans in this context. Hence, we say that we try to teach some features to an artificial worm whose nervous system is based on the data of the connectome of C. elegans.

To demonstrate this idea, we are describing one kind of experiment of using reinforcement learning on our artificial worm, namely we would like it to make a forward motion as a response to gentle touch. To represent a touch, we put some input into some sensory neurons and let the signal propagete itself through the network. By the state of the muscle cells, we may be able to say something about the motion the worm may or may note perform.

The obvious snag here is that there is no straightforward way to map the sate of the muscles onto the set of possible motions. To eliminate this issue, we used an approximation of the motion based on [14]. We imagine our worm as a four wheel vehicle, meaning that the front right, front left, rear right and rear left groups of muscles correspond to the wheels. We also make the assumption that the front wheels are mainly responsible for the direction of the motion, while the rear ones mainly give the force to move along. Exploiting the fact that a real C. elegans moves along sine waves, we consider a group of muscles being fired, it at least half of the muscles of the group are firing at a given time step, while the opposite side group is not.

We would like to learn the threshold values of the muscle groups. By doing so, we may reach that the muscle states during a simulation indicate a sine wave like motion. In this scenario, each muscle of a given group has the same threshold value, which is chosen from the set $\{3, 6, 9, 12, 15, 17, 20\}$, yielding us to the number of $7^4 = 2401$ possible threshold configurations. We consider a possible configuration of thresholds an action.

The reward depends on the state of the muscles at the end of a simulation. Using the motion approximation described above, the reward is positive and equals to the distance of the motion, if this distance is positive and the direction of the motion is straightforward. Otherwise, the reward equals to a large negative constant, if the distance is 0, and if the distance is positive, but the motion is not straightforward, the reward is negative and the bigger the direction of the movement differs from forward motion, the smaller the reward is.

We simulated two different scenario, namely anterior and posterior gentle touch. For each simulation of the former case, the input of *AVM*, *ALML* and *ALMR* is sampled from normal distribution with the same expected value and standard deviation. We imagine one step to be one simulation. After twenty thousand steps, the learning algorithm does not seem to converge.

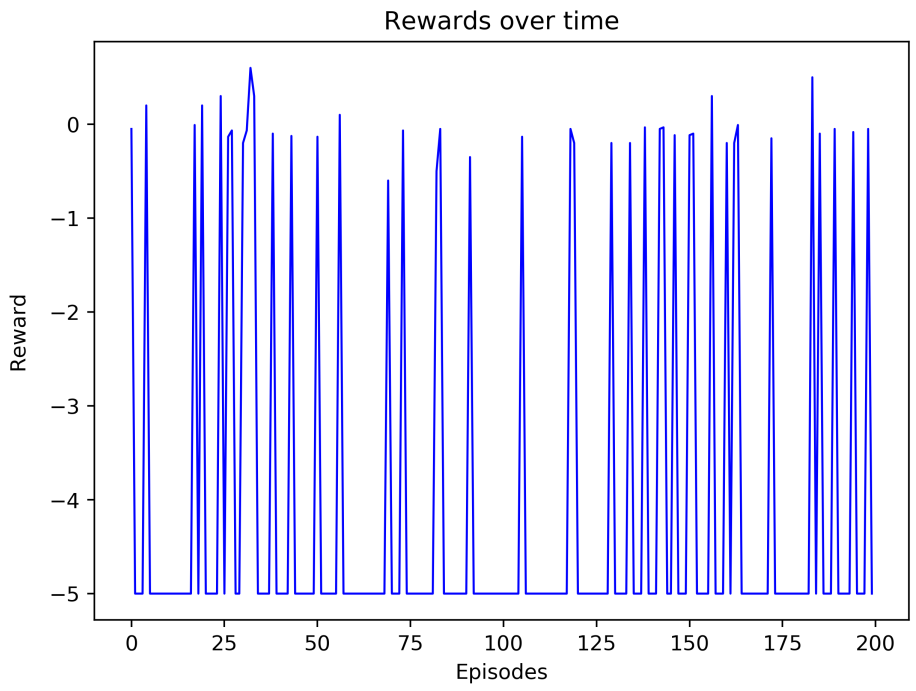
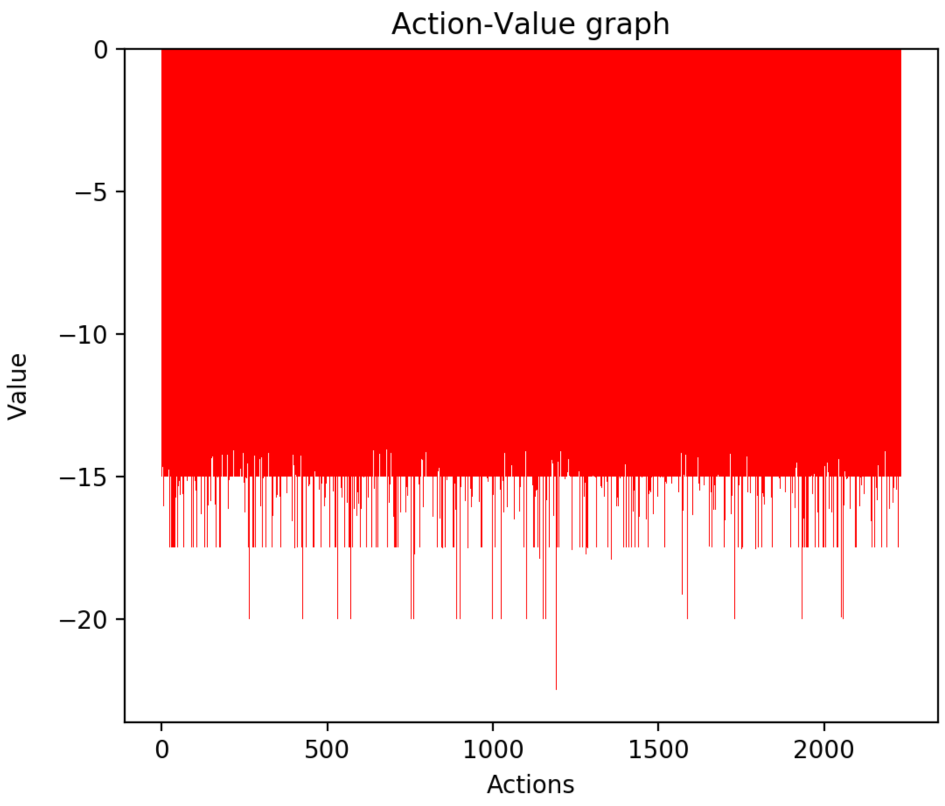


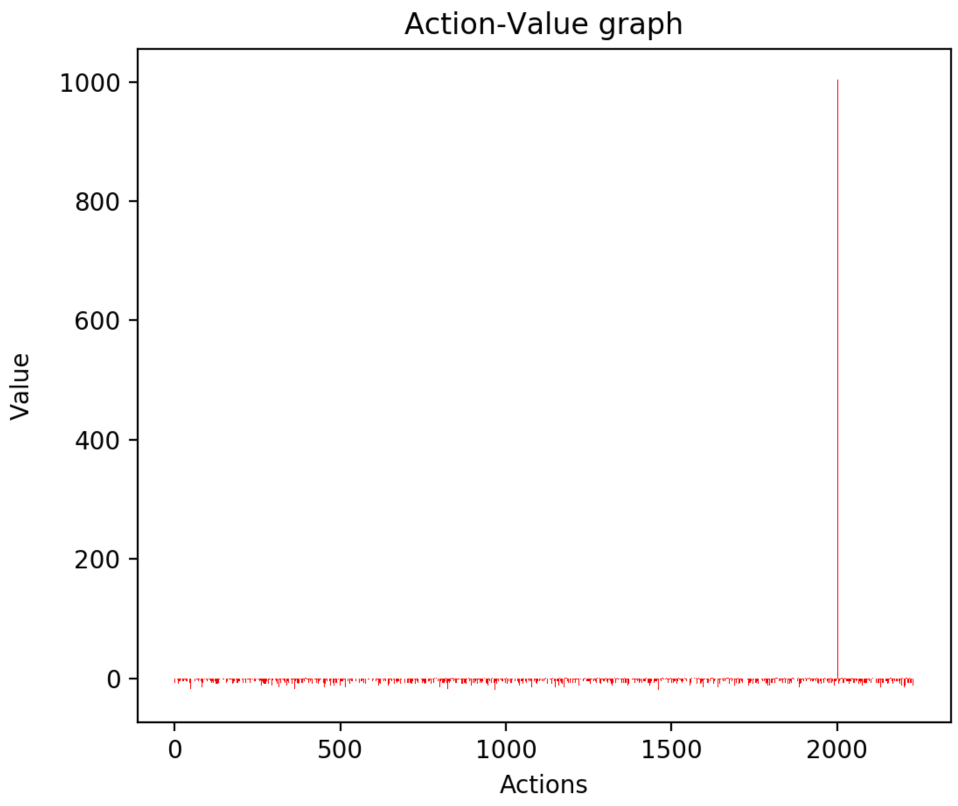
Figure 4.2: Simplified view of the rewards

The rewards of the configurations seems to show the same pattern during the whole learning process. The punishment value for not moving is -5, one can see that the worm barely moves forward. The average reward for the whole process was $-3,46$. The action-value pairs after the end of the learning process can be seen in the next figure.



This shows that basically no threshold configuration is better than any other, and also corresponds to the bad reward rate. After repeating this experiment one more time we got basically the same result, there was no sign of convergence to any configuration.

For the posterior touch case, the neurons *PLML* and *PLMR* ([24]) were fed with input generated from a normal distribution. Unlike in the anterior touch case, we experienced that the learning algorithm converged to one of the configurations.



After repeating the experiment, successful learning was achieved again, the only difference was that this time a different configuration was learnt. Both of these configurations had the same threshold value for the front right and rear right muscle groups, namely 17 and 3. As or the front and rear left groups, one of the configurations has the values of 12 and 15, while the other one has the values of 17 and 20, respectively.

One may notice that the structure of these threshold configurations are quite similar. While the relatively large difference between the threshold values of the rear groups reflects the essence of the rewarding system of this setup, one may notice that this result may not be suitable for a real-life example, as one side is much more sensitive to neural signals than the other one.

5. Discussion

In this thesis, the main control theoretic results have been introduced along with a successful application on a real-life biological system. In the final chapter, the basic terminology of reinforcement learning were given in order to apply Q-learning on our model based on the nervous system of C. elegans. While we only described one type of experiment, one can create many kind of setup in order to discover some hidden properties of the system. However, for the sake of any future work, fine tuning the attributes of the model is a fundamental task to perform successful analysis. Some of the important aspects that need to be carefully reviewed are

- the threshold values of neurons
- the input values should be fitting well with these neuron threshold values
- the fatigue rate of the muscle cells
- the possible configurations shall be detailed enough
- the correctness of the approximation for the motion vector
- the reward function should closely reflect our objective
- how many and which sensory neurons should be given external input
- any major biological misconception from our side

Bibliography

[1] https://github.com/danielracz/thesisMScPub.

[2] Neuronal wiring. http://www.wormatlas.org/neuronalwiring.html. Accessed: 2019-02-01.

[3] Stackexchange. https://math.stackexchange.com/questions/1379404/does-every-non-trivial-variety-in-mathbbrn-have-empty-interior. Accessed: 2019-05-09.

[4] William L. Brogan. *Modern Control Theory (3rd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

[5] Richard C. Dorf and Robert Bishop. *Modern Control Systems, 12th Edition*. 07 2010.

[6] Dóra Csajkó. Lecture notes on differential equations. Lecturer: Péter Simon, 2015, Faculty of Science, Eötvös Loránd University.

[7] Bryan C Daniels, William S Ryu, and Ilya Nemenman. Automated, predictive, and interpretable inference of caenorhabditis elegans escape dynamics. *Proceedings of the National Academy of Sciences*, page 201816531, 2019.

[8] Jianxi Gao, Yang-Yu Liu, Raissa M D'souza, and Albert-László Barabási. Target control of complex networks. *Nature communications*, 5:5415, 2014.

[9] Wei-Feng Guo, Shao-Wu Zhang, Qian-Qian Shi, Cheng-Ming Zhang, Tao Zeng, and Luonan Chen. A novel algorithm for finding optimal driver nodes to target control complex networks and its applications for drug targets identification. *BMC genomics*, 19(1):924, 2018.

[10] Shigeyuki Hosoe. Determination of generic dimensions of controllable subspaces and its application. *Research Reports of Automatic Control Laboratory, Faculty of Engineering, Nagoya University*, (28):p78–83, 1981.

[11] Rudolf Emil Kalman. Mathematical description of linear dynamical systems. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, 1(2):152–192, 1963.

[12] Ching-Tai Lin. Structural controllability. *IEEE Transactions on Automatic Control*, 19(3):201–208, 1974.

[13] Yang-Yu Liu, Jean-Jacques Slotine, and Albert-Lászlo Barabási. Controllability of complex networks. *nature*, 473(7346):167, 2011.

[14] Emma Marriott and Mike Yu. Examining neural behavior in c. elegans. 2016.

[15] Kazuo Murota and Svatopluk Poljak. Note on a graph-theoretic criterion for structural output controllability. *IEEE transactions on automatic control*, 35(8):939–942, 1990.

[16] E Niebur and P Erdős. Modeling locomotion and its neural control in nematodes. *Comments on Theoretical Biology*, 3(2):109–139, 1993.

[17] Ernst Niebur and Paul Erdős. Theory of the locomotion of nematodes: dynamics of undulatory progression on a surface. *Biophysical journal*, 60(5):1132–1146, 1991.

[18] Ernst Niebur and Paul Erdős. Theory of the locomotion of nematodes: control of the somatic motor neurons by interneurons. *Mathematical biosciences*, 118(1):51–82, 1993.

[19] Alex Olshevsky. Minimum input selection for structural controllability. In *2015 American Control Conference (ACC)*, pages 2218–2223. IEEE, 2015.

[20] Svatopluk Poljak. On the generic dimension of controllable subspaces. *IEEE Transactions on Automatic Control*, 35(3):367–369, 1990.

[21] Robert Shields and J Pearson. Structural controllability of multiinput linear systems. *IEEE Transactions on Automatic control*, 21(2):203–212, 1976.

[22] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*, volume 199. Prentice hall Englewood Cliffs, NJ, 1991.

[23] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[24] Gang Yan, Petra E Vértes, Emma K Towlson, Yee Lian Chew, Denise S Walker, William R Schafer, and Albert-László Barabási. Network control principles predict neuron function in the caenorhabditis elegans connectome. *Nature*, 550(7677):519, 2017.

This thesis has been created within the project of mathematical foundations of artificial intelligence 2018-1.2.1-NKP-00008.