Reachability problems for general rotor walks in graphs

Loric Duhaze-Pradines

université PARIS-SACLAY

Reachability problems for general rotor walks in graphs

Problèmes d'accessibilité des marches de rotors générales dans les graphes

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580, Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique
Graduate School : Informatique et Sciences du Numérique.
Référent : Université de Versailles-Saint-Quentin-en-Yvelines

Thèse préparée dans les unités de recherche **DAVID** (Université Paris-Saclay, UVSQ) et **LISN** (Université Paris-Saclay, CNRS), sous la direction de **Johanne COHEN**, Directrice de recherche, le co-encadrement de **David AUGER**, Maître de Conférences, et le co-encadrement de **Pierre COUCHENEY**, Maître de Conférences

Thèse soutenue à Versailles, le 06 Décembre 2023, par

Loric DUHAZÉ

Composition du jury

Membres du jury avec voix délibérative

Jean-Michel FOURNEAU Professeur des universités, DAVID, Université de Versailles-Saint-Quentin-en-Yvelines	Président
Laurent GOURVÈS Directeur de recherche, LAMSADE Université Paris Dauphine	Rapporteur & Examinateur
Kévin PERROT Maître de conférences HDR, LIS, Aix Marseille Université	Rapporteur & Examinateur
Olivier HUDRY Professeur des universités, LTCI, Télecom Paris	Examinateur
Fanny PASCUAL Maîtresse de conférences, LIP6, Sorbonne Université	Examinatrice

THESE DE DOCTORAT

NNT : 2023UPASG051

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Problèmes d'accessibilité des marches de rotors générales dans les graphes
Mots clés : Marches de Rotors, Théorie algorithmique des jeux, Complexité, Algorithmique des graphes

Résumé : Nous nous intéressons dans cette thèse aux propriétés algorithmiques d'un automate cellulaire, les marches de rotors. Ce modèle a été introduit de deux manières différentes. Tout d'abord comme une opération élémentaire d'un autre automate cellulaire : les Sandpiles qui modélisent l'effondrement d'une pile de sable lorsque celle-ci devient trop haute. Mais également, par sa ressemblance avec des modèles stochastiques très étudiés que sont les marches aléatoires. En effet, de nombreuses propriétés structurelles des marches aléatoires (temps d'atteinte, temps de couverture, etc...) sont similaires à celles de cet automate complètement déterministe qu'est la marche de rotor. Cette forme de "dérandomisation" de processus aléatoire a été la motivation principale de cette thèse. Plus précisément, une marche de rotor correspond au mouvement d'une particule sur un graphe orienté en suivant la règle suivante : au départ on fixe un ordre (une numérotation) sur les arcs sortants de chacun des sommets du graphe puis, une fois qu'on a définit la position de départ de la particule, chaque fois que cette dernière est sur un sommet, elle le quitte par l'arc de valeur la plus faible qu'elle n'a pas déjà utilisé. Bien entendu, si tous les arcs ont été utilisés, on redémarre avec l'arc de plus faible valeur. Il existe une multitude de problèmes d'accessibilité sur les rotors dont nous nous appliquons à faire une liste dans cette thèse. Nous donnerons également des résultats de complexité pour certains d'entre eux. Puis nous nous intéresserons à un problème d'accessibilité particulier : ARRIVAL. Si l'on considère un graphe avec des puits tel qu'il existe un chemin orienté entre chaque sommet du graphe et au moins l'un de ces puits, une marche de rotor se termine forcément. Hélas, le nombre d'étapes avant que ce processus ne termine peut être exponentiel. En 2017, Dorhau et al. ont présenté un problème, nommé ARRIVAL, qui est de savoir si la particule finit bien sa course dans un puits donné. Ils ont montré qu'il appartenait aux classes de complexité NP et co-NP. Etant donc un bon candidat à être résolu par un algorithme polynomial, nous nous intéressons à ce problème sur une sous-classe de graphe pour laquelle le nombre d'étapes du processus peut être exponentiel : les Tree-like multigraphes. Il s'agit de multigraphes donc le graphe non-orienté sous-jacent est un arbre. Dans ce contexte, nous avons pu montrer que ce problème pouvait être résolu en temps linéaire et même étendre ces résultats à des versions décisionnelles du problème ARRIVAL connues pour être respectivement NP-complète et PSPACE-complète.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: Reachability problems for general rotor walks in graphs
Keywords: Rotor Walks, Algorithmic game theory, Computational complexity, Graph algorithms

Abstract: In this thesis, we focus on the algorithmic properties of a cellular automaton known as rotor walks. This model has been introduced in two distinct ways. Firstly, as a fundamental operation within another cellular automaton known as Sandpiles, which models the collapse of a sand pile when it becomes too high. Secondly, due to its resemblance to well-studied stochastic models, such as random walks. Indeed, numerous structural properties of random walks (hitting times, cover times, etc.) are analogous to those of this completely deterministic automaton called the rotor walk. The main motivation for this thesis stems from this "derandomization" of a random process. More precisely, a rotor walk corresponds to the movement of a particle on a directed graph following the following rule: initially, an order (a numbering) is fixed on the outgoing arcs of each vertex of the graph. Once the starting position of the particle is defined, each time it is on a vertex, it leaves through the arc with the lowest value that it has not already used. Of course, if all arcs have been used, the process restarts with the lowest value arc. There is a multitude of accessibility problems on rotors, and we aim to compile a list of them in this thesis. We also provide complexity results for some of these problems.
Subsequently, we turn our attention to a specific accessibility problem: ARRIVAL. Considering a graph with sinks such that there is a directed path between each vertex of the graph and at least one of these sinks, a rotor walk inevitably terminates. Unfortunately, the number of steps before this process concludes can be exponential. In 2017, Dorhau et al. introduced a problem called ARRIVAL, which seeks to determine if the particle successfully reaches a given sink. They demonstrated that it belongs to the complexity classes NP and co-NP. Being a strong candidate for polynomial algorithm resolution, we investigate this problem on a subclass of graphs where the step count of the process can be exponential: Tree-like multigraphs. These are multigraphs whose underlying undirected graph is a tree. In this context, we show that this problem can be solved in linear time, extending these results to decision versions of the ARRIVAL problem, known to be respectively NP-complete and PSPACE-complete.

Titre : Problèmes d'accessibilité des marches de rotors générales dans les graphes

Synthèse : Cette thèse présentent les travaux que nous avons effectués concernant les propriétés algorithmiques des marches de rotors ainsi que l'étude d'un problème particulier sur les marches de rotor : ARRIVAL. Le *problème ARRIVAL*, défini par Dohrau *et al.* dans [31], est un problème de décision sur un graphe orienté avec des puits où chaque sommet qui n'est pas un puits a exactement deux arcs sortants. L'objectif est de décider si une particule se déplaçant sur ce graphe selon une marche de rotors atteint un certain sommet puits (voir Figure 1.1 pour un exemple). La règle de déplacement est la suivante : chaque sommet a un arc sortant initial "rouge" que la particule peut emprunter, et une fois qu'une particule emprunte un arc rouge, l'arc rouge du sommet qu'elle vient de quitter est mis à jour vers l'autre arc sortant de ce sommet. Cette règle déterministe garantit que la particule traversera les deux arcs presque le même nombre de fois, comme cela serait observé en moyenne avec une marche aléatoire. Un mouvement est dit *légal* s'il y a une particule sur le sommet où l'opération de déplacement a été effectuée. Inversement, un mouvement est *illégal* s'il a lieu à un sommet sans particule. Le problème ARRIVAL, dans sa définition originale, ne permet que des mouvements légaux. Dans les chaînes non-orientées à n sommets, le train atteint un puits avec au plus O(n^2) mouvements légaux, fournissant ainsi une solution au problème ARRIVAL. Cependant, dans des cas plus généraux, le nombre d'étapes nécessaires pour que le train atteigne un puits peut être exponentiel, comme illustré dans la Figure 3.2 du Chapitre 3. Néanmoins, il a été démontré dans [31] que le problème ARRIVAL appartient à la classe de complexité **NP∩co-NP**. Pour étudier ce problème, notre approche est la suivante.

Nous avons choisi de développer un nouveau cadre unifié pour l'analyse des problèmes liés aux marches de rotors, en autorisant la présence de plusieurs particules mais également de particules négatives (antiparticules, introduites dans le chapitre Chapitre 2). Ce cadre unifié nous a permis de généraliser divers résultats déjà connus dans d'autres contextes et de développer une compréhension plus approfondie des marches de rotor dans les graphes. Nous avons notamment pu montrer que la propriété d'unicité des configurations de rotors et de particules obtenues après un routage légal maximal pouvait être étendue au cas des routages non-légaux. Dans ce dernier cas, la notion de maximalité du routage est vue comme le fait de rassembler toutes les particules et antiparticules sur les puits du graphe et l'unicité concerne seulement la configuration de particules et pas la configuration de rotors.

Ensuite, nous nous intéressons au problème ARRIVAL comme un problème d'accessibilité, à savoir, si l'on appelle s le puits cible, et qu'on suppose que la particule est initialement à la position v avec la configuration ρ, existe-t-il une configuration de rotor ρ' telle qu'il existe un routage légal depuis la configuration de rotor-particule (ρ, v) jusqu'à la configuration de rotor-particule (ρ', s) ? Ce faisant, nous examinons divers problèmes d'accessibilité et leurs liens avec cette vision d'ARRIVAL au sein de notre cadre unifié. Notre objectif est d'obtenir une compréhension plus approfondie du problème ARRIVAL et des facteurs qui le rendent complexe sur des graphes orientés arbitraires (comme détaillé dans le Chapitre 3). En particulier nous présentons plusieurs variantes du problème ARRIVAL et nous montrons qu'elles appartiennent toutes à **NP∩co-NP**. Nous montrons notamment que la complexité du problème ARRIVAL ne réside pas dans la légalité des opérations de routage.

Enfin, nous avons examiné une sous-classe de graphes, à savoir les treelike-multigraphes (graphes orientés tels que le graphe non-orienté sous-jacent est un arbre), pour lesquels nous présentons un algorithme (présenté dans le Chapitre 4) permettant de résoudre le problème ARRIVAL en temps polynomial. Cet algorithme repose sur le calcul du nombre de fois que la particule peut voyager d'un sommet v à un sommet v' avant d'être "aspirée" par un des puits du graphe. De manière similaire aux marches aléatoires avec un ou deux joueurs (appelées processus de décision de Markov, voir [74]) et aux jeux stochastiques (voir [79]), nous avons exploré certaines variantes décisionnelles d'ARRIVAL sur les treelike-multigraphes, démontrant qu'elles peuvent également être résolues en temps polynomial.

Contents

Remerciements

This thesis is a detailed summary of the different subjects and studies we have made trough these four years of work. I am deeply grateful to Yannis Manoussakis for his unwavering support and motivation as my supervisor throughout this journey. I would also like to express my gratitude to Johanne Cohen for stepping in as the new supervisor and providing invaluable guidance and rigor to ensure the progress of this thesis. Special thanks go to David Auger and Pierre Coucheney for their valuable advice, countless brainstorming sessions, and their continuous support even during times when my motivation wavered. I would also like to express my heartfelt gratitude to my parents for fostering my curiosity in various scientific subjects, for their unwavering support, and for encouraging me to pursue my passion. And finally I would like to extend my thanks to my partner, who has been a constant source of encouragement and assistance throughout my thesis journey.

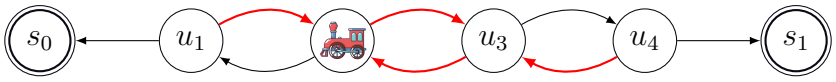
1 - Introduction

Reachability problems are typically formulated as follows: Given a dynamic system and an initial state of the system, is it possible for the system to transition to a desired target state? These types of problems gained prominence in the 1970s with the emergence of the Vector Addition System Reachability Problem (e.g., references such as [66, 11, 58, 76]) and the Petri Net Reachability Problem (e.g., references like [46, 64]). These two problems have given rise to various models encompassing complex reachability problems, spanning fields such as network routing [84], software verification [26], manufacturing supply chains [42], transportation planning [67], and aircraft collision avoidance [54]. As a result, finding efficient solutions to these problems is a crucial concern in the realm of science and technology. In this document, our measure of the efficiency of a solution (i.e., the difficulty of a problem) relies on the computational complexity class to which the problem belongs. The definitions of the various complexity classes mentioned in this document can be found in Appendix.

Like simple stochastic games (see [5]), in this thesis, we study a reachability problem in finite graphs that falls into the complexity class **NP**∩**co-NP**, and to date, no polynomial algorithms have been discovered to solve them. However, these problems are promising candidates for membership in the **P** class. This motivated our choice to investigate the ARRIVAL problem, which also falls into the **NP**∩**co-NP** category and lacks a known polynomial-time solution algorithm. Furthermore, the ARRIVAL problem is interesting as it is a deterministic analog of random walks in graphs.

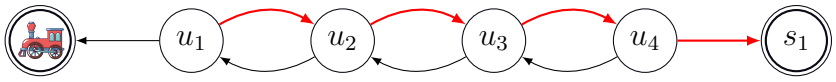
1.1 . Thesis Overview

The *ARRIVAL problem*, defined by Dohrau *et al.* in [31], is a decision problem on a directed graph with sinks where every vertex that is not a sink has exactly two outgoing arcs. It consists in deciding whether a particle moving on this graph with specific rules (rotor walk) reaches a certain sink vertex (see Figure 1.1 for an example). The movement rule is as follows: each vertex has an initial "red" outgoing arc that a particle can travel through, and once a particle travels through a red arc, the red arc of the vertex it just left is updated to the other outgoing arc of that vertex. This deterministic rule ensures that the particle will travel through both arcs almost the same number of time, as would be observed in average with a random walk. A move is said to be *legal* if there is a particle on the vertex where the move operation was proceeded. Conversely, a move is *non-legal* if it proceeds at a vertex having no particle. The ARRIVAL problem only permits legal movements.



(a) The train represents the particle on this directed graph. Red arcs are the currently available paths for the train. Here, it can only move up to u_3.

(b) When moving to u_3, the available paths for the train are modified. Note that at each step the train only has one available path from its position.



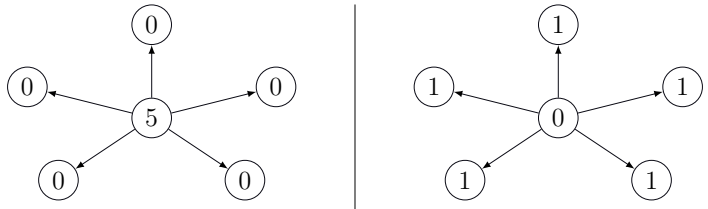
(c) By proceeding enough movements, the train will eventually reach the sink s_0. Hence, since this is a deterministic process, it will not reach s_1.

Figure 1.1: Instance of an ARRIVAL problem and the corresponding legal move sequence where the train reaches s_0.

In path graphs with n vertices (as in the example), the train reaches a sink with at most $O(n^2)$ legal movements, thus providing a solution to the ARRIVAL problem. However, in more general cases, the number of steps required for the train to reach a sink may be exponential, as illustrated in Figure 3.2 in Chapter 3. Nevertheless, it has been demonstrated in [31] that the ARRIVAL problem falls within the complexity class **NP∩co-NP**. To study this problem, our approach was the following. We first had a look at a subclass of graphs, namely treelike multigraphs, for which we proved the existence of a polynomial time algorithm solving the ARRIVAL problem (presented in Chapter 4). Similarly to random walks with one or two players (so called Markov Decision Process (see [74]) and Stochastic Games (see [79])), we explored certain decisional variants of ARRIVAL on treelike multigraphs, demonstrating that they also can be solved in polynomial time. Next, we opted to develop a new unified framework for the analysis of rotor-routing related problems, accommodating the presence of multiple particles as well as negative particles (antiparticles) (introduced in Chapter 2). This unified framework has allowed us to generalize various results that were already known in different contexts and to develop a deeper understanding of rotor walks in graphs. And lastly, we returned to our initial objective, examining a range of reachability problems and their connections to ARRIVAL within this unified framework. Our aim was to gain deeper insights into the ARRIVAL problem and the factors that make it challenging on arbitrary directed graphs (as detailed in Chapter 3).

1.2 . Related Works

While most results concern rotor-routing, this thesis will introduce some concepts on particle configurations closely related to a more algebraic-focused model: the Sandpiles model. Let us take a little time to present this model and its re-



(a) Graph with 5 sand grains on the central vertex.

(b) One sand grain has been moved on each exterior vertex after the central vertex collapsed.

Figure 1.2: Detail of the collapsing rule for sandpiles (see [16]). When there are a number of sand grains superior or equal to the outdegree of a vertex, this vertex collapses.

lations with the rotor-routing model. Both rotor-routing and sandpiles model are automatons relying on a graph structure with some particles (e.g. chips, tokens, ...) moving on this graph. The difference comes from the rules of movement.

Sandpiles

The Sandpiles model corresponds to a phenomenon that occurs in nature: the piles of sand collapsing when they are too high (rule is detailed in Figure 1.2). This collapse is an example from a vast family of phenomenons that have the property of critically self-organize.

The Sandpiles model, presented initially under the name Bak–Tang–Wiesenfeld (BTW) model in [10], tries to give some theoretical rules about whether a pile of sand grains collapses depending on its height. Since this model is made to give a better understanding of a real natural phenomenon, particles move on the graph only if there is a lot of particles on the same vertex, in which case the vertex collapses and particles are dispatched between its different neighbours (such a collapse is called a *firing* in the literature and in this document). This model has some strong algebraic properties studied first in physics since 1990 [27, 30] from where it inherits its current name, the Abelian Sandpiles Model. Rapidly, mathematicians presented an abelian group structure (e.g., the Sandpile Group) on some particular elements of a theoretical sandpile: recurrent states. Following this popularity, several authors presented papers on the structure of the sandpile group on specific graph classes. In 1999, Cori and Rossin established in their work [24] that for planar graphs, the Sandpiles group is isomorphic to that of its dual graph. Subsequently, Toumpakari made significant progress by computing the structure of the Sandpiles group for regular trees and proposing a conjecture suggesting that the Sandpiles group for such graphs can be decomposed into a product of the Sandpiles groups of its subtrees. This conjecture was later confirmed by Levine in [60]. Similar investigations were carried out by Chen and Schedler on thick trees with loops in [21]. In 2011, Levine extended the analysis by computing the structure of the Sandpiles group for De Bruijn graphs and Kautz graphs, as documented in [59]. Furthermore, in 2008, Holroyd and colleagues provided a comprehensive survey [50]

of existing results on the Sandpiles model and highlighted the connections between the Sandpiles model and the rotor-routing model. Shortly after, a new problematic emerged: computing the identity element of the sandpile group (see an example for a square grid on Figure 1.3) when describing the whole structure is too hard (e.g. [53, 32]) In [57], it was demonstrated that on a rectangular grid, the identity element can be decomposed into three parts, with the central area being uniform, as illustrated in Figure 1.3. This decomposition of the identity element has been a subject of interest. Researchers have also explored the shape of the identity element for other types of grids, such as triangular and hexagonal grids, as well as various tilings, as discussed by Fersula *et al.* in [36]. Additionally, Alfaro and colleagues successfully computed the identity element of the cone of a regular graph by solving an integer linear programming problem in their work [1].

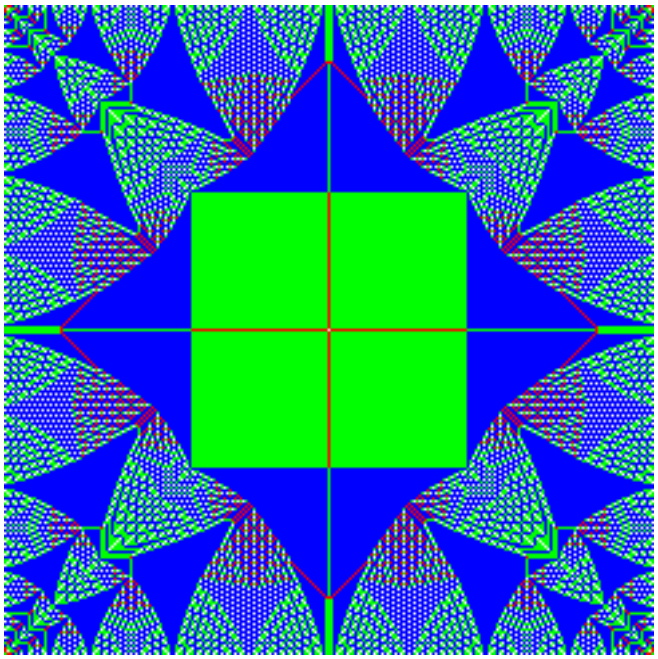


Figure 1.3: The identity element of the sandpile group of the 128x128 square grid (image from [50]) with sinks on the edges of the square. The color scheme is as follows: white=0 chips, red=1 chip, green=2 chips, and blue=3 chips.

The mechanism described by the sandpiles model has been used for many other purposes. We can cite the Engel Machine (illustrated in Figure 1.4), whose purpose is to make conversions for numbers written in different bases. This machine is anterior to the use of the term "sandpiles", but it relies on the same principle. Indeed, in both cases, when the number of elements on a particular vertex is superior or equal to the number of exits, one element is sent through each exit. We will define this process as **firing** shortly after in this document. An example of utilization of an Engel Machine is shown in Figure 1.4.

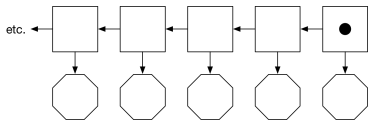


Figure 1.4: Vertices of this Engel Machine can be separated in two sets, the square vertices of degree 2 and the non-square vertices which are sinks (all particles arriving on them will stay there forever). We put a single particle on the rightmost vertex.

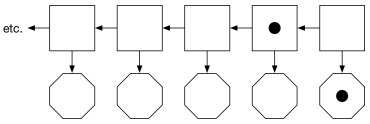


Figure 1.5: By adding a particle on the rightmost square vertex, the number of particles on it is now 2 making this vertex topples/fires. This operation sends one chip to each neighbour of this vertex. We see here that the number of particles on the square cases are the binary writing of the total number of particles put on the rightmost vertex.

Sandpiles have also been studied from a reachability point of view (which we will present in Chapter 3). The main result of [43] is the undecidability of this problem for infinite graphs. It has also been studied on general finite digraphs ever since and proved to be in **co-NP** (e.g. [52]).

Dollar Game

A version of the Sandpiles model where vertices are allowed to borrow particles from their neighbour quickly emerged: the Dollar Game [14]. In the dollar game, each particle represents one unit of money, and each node represents a person who can either have exceeding money (a positive number of particles on the vertex) or have debt (a negative number of particles on the vertex). A negative number of particles on a vertex can be seen as an amount of negative particles (or antiparticles). Each person can either give one unit of money to each of their neighbors or borrow one from them. The term "Dollar Game" gets its name from the objective of clearing each vertex's debt through lending and borrowing operations. The primary result from [14] asserts that if a solution exists for a given instance of the Dollar Game, there is a greedy algorithm that can solve it. This generalized version of Sandpiles served as one of the main inspiration for developing a generalized theory around rotor routing which constitutes a significant contribution of this thesis. Certainly, the inclusion of negative particles, which we'll refer to as antiparticles, significantly changes the rotor-routing model. This is because the legitimate movement of a particle can be halted when it reaches a vertex with a negative particle count. In Figure 1.1, if u_3 has a negative particle count, the train stops immediately after a single legal move.

Rotor-routing Model

In 1996, Priezzhev and al. [72, 71] presented the first version of what will later be the rotor routing model under the name Eulerian Walker. In a directed graph, we fix a total order on the outgoing neighbors (or outgoing arcs) of each non-sink vertex. Then, a vertex with a particle on it topples (i.e., is routed) and the particle moves towards one of this vertex's neighbors following the previously set order. This can be seen as an elementary step of a firing operation of the Sandpiles model, and it is why the links between these two models have brought so much interest [50]. This process was rediscovered several times, in mathematics under the name rotor-routing [75, 33] and under the name ant-patrolling network in distributed algorithmics [85]. From its close relation to the Sandpiles model, this model also has some interesting algebraic properties. One of the most studied ones is the group action of the abelian group of sandpiles on the spanning in-arborescences of a digraph through rotor-routing [56, 80, 19]. In this thesis, we will present and construct the equivalence classes of rotor-routing model using a different approach. While there is a connection between sandpiles and rotor-routing that explains the existence of these equivalence classes, we will explore alternative methods to define and understand them within the context of our research.

Another interesting question about the rotor-routing model is the derandomization of random walks. Indeed, this process can be seen as a deterministic version of a random walk, so it shares many properties with its stochastic variant, the Markov chain. Notably, Holroyd and Propp proved in [49] that several natural statistics of markov chains are approximated by rotor walks. One can mention that the cover time of the vertices for trees coincides in average for random walks and rotor walks [33].

The rotor-routing model has also been studied for its structural properties. Namely, consider a graph with sink vertices. In order to compute the order in which the sinks will be reached if we repeat the rotor-routing process starting from the same vertex, Giacaglia *et al.* [41] introduced the rotor class equivalence that we will study in detail in Chapter 2 .

In parallel, a series of papers starting in 2017 (e.g., [31, 40, 39, 63]) were published where authors seemed unaware of previous results concerning rotor walks. Their work focuses on a new complexity question about reachability problems in so-called *switching graphs* which are in fact a special kind of rotor graphs. They tried to answer the following question: which sink is reached by the particle if we know the starting configuration and the starting position of a particle? This problem is originally known as the ARRIVAL problem and it has been demonstrated to be part of both **NP** and **co-NP** in a study by Dohrau *et al.* in 2017 [31]. Moreover, there have been advancements in improving the upper bounds for this problem. The ARRIVAL problem is also in **UP** (Unique Polynomial Time) and **co-UP**, indicating the existence of efficient verifiers that can accept unique proofs (e.g., proof in [40]). A search version of the ARRIVAL problem has been introduced and proven to be

in **PLS** (Polynomial Local Search). Furthermore, this version has been demonstrated to be in **CLS** (Combinatorial Local Search), which is the intersection of **PLS** and **PPA** (Polynomial Parity Arguments on Directed graphs) and finally in **UniqueEOPL** (Unique Equilibrium Oriented Polynomial Local Search), a complexity class capturing total search problems with a unique solution. An algorithm with subexponential complexity, specifically in time $O(2^{\sqrt{n}\cdot\log n})$, has been proposed for the ARRIVAL problem on graphs with n vertices in [39]. On the other hand, a lower bound has been established in [63], proving that the ARRIVAL problem is **CC-Hard** (Counting Class Hard) and **PL-Hard** (Polynomial Local Search Hard). The study of this problem in our new unified framework and for a particular class of graphs will be another major part of the work developed in this thesis.

Please, note that the "sand grains" and "trains" described in the introduction will be replaced by "particles" in the core document to make the definitions and properties clearer.

1.3 . Document organization and contributions

We give here a brief summary and the main results of each chapter.

- Chapter 2 presents a new general framework unifying rotor-routing model, sandpiles model and dollar game with and without the legality constraint. This framework was initially developed at the beginning of our work on a generalized version of the ARRIVAL problem with strong algebraic structural properties. Our expectation was that these structural properties could be transferred to the ARRIVAL problem itself. Therefore, we presented rotor equivalence classes, particle configuration equivalence classes, and rotor particle equivalence classes. It also introduces novel and expanded properties for the unified rotor-routing model applied to strongly connected and stopping graphs. In particular, we prove a general asymptotic result on the finiteness of legal routing sequences with any initial particle configurations (positive or negative).

  This framework has been used in a publication at RP2023 [6].

- In Chapter 3, we present the ARRIVAL problem and its various versions, including scenarios involving multiple particles and scenarios without the legality constraint. One of the key problems we define is **SP-ARRIVAL**, which is exactly the problem originally presented in [31], but adapted to our framework. We prove that they all belong to **NP∩co-NP**. Next, we focus on various legal or non-legal other reachability problems in the context of strongly connected graphs and stopping graphs. We present some polynomial reductions between some of these problems and variants of the

ARRIVAL problem. In particular, we define a generalized version of the ARRIVAL problem where non-legal routing sequences are permitted. And we show that this version is equivalent to a variant of the ARRIVAL problem with several particles on strongly connected and stopping graphs. These results are currently in the drafting phase and will be submitted to a journal for publication.

- Chapter 4 focuses on the SP-ARRIVAL problem. We introduce a class of graphs, namely treelike multigraphs, where we demonstrate that solving **SP-ARRIVAL** can be accomplished in polynomial time. Furthermore, we extend this result to decisional variations of **SP-ARRIVAL**, considering scenarios with one or two players. We prove that on treelike multigraphs, these problems can also be solved in polynomial time, even though they are **NP-Complete** and **PSPACE-Complete** on general directed graphs. These findings have been published at MFCS2022 [3].

Throughout this document, when we mention properties that are drawn from existing literature, we provide references to the original papers where these properties were introduced. Additionally, it's important to note that a portion of our work involves recombining and generalizing existing findings. In such cases, we include annotations to reference the original works from which these ideas were derived.

2 - General Framework of Rotor Routing

In this chapter, we will introduce the concept of the "rotor-routing" operation and discuss its essential properties for studying various problems covered in Chapters 3 and 4. The framework developed in this chapter has been used in a work published at RP2023 [6].

In traditional rotor-routing frameworks, vertices can only be routed if they have a positive number of particles on them (e.g., [81, 77, 41]). However, we are interested in exploring reachability problems, (e.g., a generalized version of the ARRIVAL problem) where vertices can be routed even if they have a negative particle count and routing operations are not necessarily legal. This is why we introduce in this chapter a unified framework for rotor-routing problems.

The organization of this chapter is as follows. In Section 2.1, we provide formal definitions for rotor-routing mechanics and describe our routing rule, which determines how particles move based on the current configuration. Next, in Section 2.2, to capture the movement of particles in a routing sequence, we present the concept of *routing vector* introduced in [81], and we show that it can be used as a certificate of the ending configuration of a routing sequence. This vector succinctly represent the displacement of particles during each routing operation, making it easier to analyze and understand the system's behavior. In Section 2.3, we introduce an operation called *cycle-push*, which is a well-known technique in the field. The cycle-push operation represents a specific type of routing sequence that captures the movement of a particle along a circuit in the graph. By using the cycle-push operation, we can establish equivalence relations between different rotor configurations. This means that certain rotor configurations can be transformed into one another through the application of cycle-push. We also present some properties on particle configurations equivalence classes. In addition to the traditional rotor-routing framework, we introduce, in Section 2.4, a more intuitive variant called the *rotor walk*, which is a routing sequence with a single particle originally introduced in [72]. Rotor walks allow us to express general properties in a simpler and more intuitive manner. Through the concept of rotor walks, we aim to provide a clearer and more intuitive understanding of the dynamics and behavior of rotor configurations and their interactions with the underlying graph structure. Finally, in Section 2.5, we introduce the notion of *legality* in rotor-routing. Legality imposes constraints on the routing sequences, specifying which routings are allowed or forbidden. We discuss the concept of legality and present a key result that motivates our exploration of reachability problems in Chapter 3. Furthermore, we examine characteristic properties of legal routing sequences, focusing on how they depend on the graph's topology.

2.1 . Basic Definitions

Graph Statement

Unless otherwise stated, we consider a directed multigraph $G = (V, \mathcal{A}, h, t)$ where V is a finite set of *vertices*, $\mathcal{A}$ is a finite set of *arcs*, and h (for *head*) and t (for *tail*) are two maps from $\mathcal{A}$ to V that define the incidence between arcs and vertices. For a given arc $a \in \mathcal{A}$, vertex $h(a)$ is called the head of a and $t(a)$ is the tail of a. For simplicity, we typically denote such an arc by $(t(a), h(a))$.

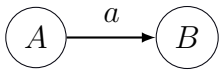


Figure 2.1: a is an arc with tail $t(a) = A$ and head $h(a) = B$.

For the sake of clarity, we consider only graphs without arcs of the form $h(a) = t(a)$ (i.e., loops). However, our results would still hold if we were to allow such arcs. Note that multigraphs can have multiple arcs with the same head and tail. Let $u \in V$ be a vertex, we denote by $\mathcal{A}^+(u)$ (resp., $\mathcal{A}^-(u)$) the multiset of arcs $a \in \mathcal{A}$ with tail u (resp., with head u). A vertex s such that $\mathcal{A}^+(s) = \emptyset$ is called a *sink*.

Let $\Gamma^+(u)$ (resp., $\Gamma^-(u)$) be the subset of vertices $v \in V$ for which there is an arc $a \in \mathcal{A}$ with $h(a) = v$ and $t(a) = u$ (resp., $h(a) = u$ and $t(a) = v$). A graph is said to be *simple* if for all $u \in V$ we have $|\mathcal{A}^+(u)| = |\Gamma^+(u)|$.

What we call a *circuit* in this document is a sequence of arcs $a_0, a_1, \cdots, a_{k-1}$ such that for $0 \leq i \leq k$ we have $h(a_i) = t(a_{i+1 \mod k})$.

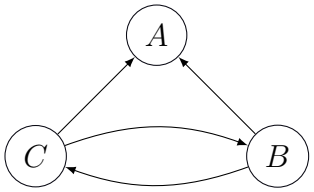
Laplacian and Period Vector

Now, we define the Laplacian Matrix of a graph [13] which is a square matrix that captures important properties of the graph's structure. The diagonal elements of the matrix represent the degree of the vertices (the number of edges connected to each of them), while the off-diagonal elements represent the connections between vertices. Next, we introduce the notion of a "period vector" which will play a key role in Section 2.2. The period vector is a mathematical representation that describes certain repetitive patterns within a graph.

Definition 2.1.1 (Laplacian Matrix)**.** *The* Laplacian matrix *of a graph G is denoted by L_G. It is defined by $L_G(u,u) = -|\mathcal{A}^+(u)|$, and $L_G(u,v) = |\mathcal{A}^+(u,v)|$ if $u \neq v$, where $\mathcal{A}^+(u,v)$ denotes the set of outgoing arcs of vertex u with head v.*

Note that, for any row of L_G, the sum of the elements of this row is 0.

We introduce a specific class of graphs, which will be one of the two cases studied in this document.



$$L_G = \begin{pmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{pmatrix}$$

Figure 2.2: Graph G and its Laplacian Matrix.

Definition 2.1.2 (Stopping Graph [1])**.** *A graph $G = (V, \mathcal{A}, h, t)$ is said to be* stopping *if for every vertex $v \in V$, either v is a sink vertex or there exists a directed path from v to some sink s.*

Recall that in a graph G, a sink component refers to a strongly connected component that does not have any outgoing arcs in $\mathcal{A}$. We call *transient components* the connected components obtained by removing sink components from the graph as illustrated in 2.3. Remark that transient components are characterized by the fact that they have at least one outgoing arc directed towards another transient component or a sink component. And, from [68], we have that any directed multigraph can be decomposed in polynomial time into sink components and transient components.

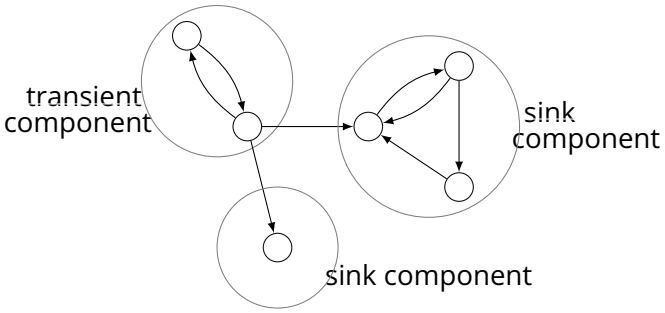


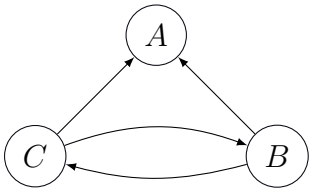
Figure 2.3: Decomposition into sink components and transient components.

Hence, without loss of generality, we restrict our study to strongly connected graphs (which are the sink components) and stopping graphs (which are the transient components).

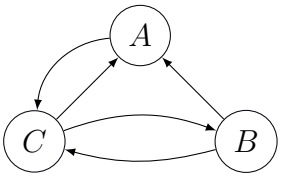
First, we introduce the classical definition of a *period vector* for a graph. This concept will be further interpreted in terms of routing in the subsequent sections of this document.

Definition 2.1.3 (Period vector of a graph)**.** *A* period vector p *for a graph G is a vector $p \in \mathbb{N}^{|V|}$, that satisfies the equation $p \cdot L_G = 0|_{|V|}$, where L_G is the Laplacian matrix associated with G and $0|_{|V|}$ is the vector of zero of length $|V|$.*

[1] As any other graph we consider, stopping graphs are directed graphs.



$$p = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$$



$$p = \begin{pmatrix} 3 & 1 & 2 \end{pmatrix}$$
$$p = \begin{pmatrix} 9 & 3 & 6 \end{pmatrix}$$
$$\dots$$
$$p = \begin{pmatrix} 6 & 2 & 4 \end{pmatrix}$$

Figure 2.4: Example of period vectors for two graphs, one that is stopping and one that is strongly connected.

Definition 2.1.4 (Primitive period vector of a graph)**.** *A period vector p of a graph G is said to be* primitive *if there is no non-trivial common divisor between the entries.*

Remark. *We will present later in detail the firing operation (i.e., routing a vertex a number of times equals to its outdegree). Given a period vector p, by proceeding a firing operation on each vertex v a number of times equals to $p(v)$, the particle configuration remains the same, hence the name "period vector".*

In Figure 2.4, the primitive period vector of the second graph is $(3\ 1\ 2)$.

Proposition 2.1.5 (Proposition 4.1 of [15])**.**

1. *The primitive period vector p of a strongly connected graph exists and is unique. It is strictly positive and all period vectors t of strongly connected graphs are of the form $t = p \cdot k$ with $k \in \mathbb{Z}$.*
2. *Period vectors of a general digraph with k sink components (strongly connected components) are exactly vectors of the form $\Sigma_{i=1}^{k} \lambda_i p_i$ where $\lambda_i \geq 0$ and p_i is the primitive period vector of size $|V|$ of the sink component C_i with 0 on any entry that does not belong to C_i.*

Following result is a direct implication of Proposition 2.1.5 but it has not been stated before.

Corollary 2.1.6 (Primitive Period vector of a stopping graph)**.** *The primitive period vector of a stopping graph is unique and is $0|_{|V|}$.*

Proof. From the fact that sink vertices are the only sink components of a stopping graph and their primitive period vector is 0, by Proposition 2.1.5 we have that the primitive period vector of a stopping graph is unique and is $0|_{|V|}$. □

Furthermore, we have the following complexity result regarding the computation of a primitive period vector.

Proposition 2.1.7 (Proposition 1.2 of [81])**.** *The primitive vector of a general digraph can be computed in polynomial time.*

2.1.1 . Rotor Routing Definitions

This subsection introduces the concept of a *routing sequence* along with the various operators required to define it. In our rotor-routing framework, in contrast to the existing literature, we introduce the concept of antiparticles and the associated operations required to move them. We treat a particle configuration as a balance between particles and antiparticles on each vertex, essentially represented by an integer value. We also examine the distinctions between the routing rule typically employed in the literature and our version of this routing rule.

Let $G = (V, \mathcal{A}, h, t)$ be a multigraph.

2.1.1.1 Rotor Graph and configurations

Definition 2.1.8 (Rotor Order)**.** *Let $v \in V$ be a vertex. A* rotor order *at v is a circular permutation on the outgoing arcs of v, namely an operator denoted by θ_v that satisfies the following properties:*

- $\theta_v : \mathcal{A}^+(v) \to \mathcal{A}^+(v)$*;*
- *For all $a \in \mathcal{A}^+(v)$, the orbit $\{a, \theta_v(a), \theta_v^2(a), \dots, \theta_v^{|\mathcal{A}^+(v)|-1}(a)\}$ of arc a under θ_v is equal to $\mathcal{A}^+(v)$, where $\theta_v^k(a)$ denotes the composition of θ_v applied to arc a exactly k times*

Thus, θ_v is a bijection and we have $\theta_v^{-1}(a) = \theta_v^{|\mathcal{A}^+(v)|-1}(a)$ for any $a \in \mathcal{A}^+(v)$.

Note that every arc in $\mathcal{A}^+(v)$ appears exactly once in any orbit of θ_v. We will now incorporate the operator θ_v into our graph structure in the following way.

Remark (Rotor order representation)**.** *Please note that in the rest of this thesis, rotor order will be depicted in Figures by a red cyclic arrow similar to the one of Figure 2.5.*

Definition 2.1.9 (Rotor Graph)**.** *A* rotor graph *G is a (multi)graph $G = (V, \mathcal{A}, h, t)$ together with the following components:*

- *a vertex partition $V = V_0 \cup S_0$, where V_0 consists of the vertices that have at least one outgoing arc and S_0 is the possibly empty set of sink vertices, which are the vertices with outdegree zero*

Figure 2.5: Rotor order on a graph. In the examples presented in this document, we will consider the same planar rotor order (clockwise or anticlockwise) on the outgoing arcs of each vertex for better understanding. But all results are constructed with general rotor orders as defined in Definition 2.1.8.

- *a mapping θ from $\mathcal{A}$ to $\mathcal{A}$ such that for each $u \in V_0$, θ_u is a rotor order at u where θ_u is the mapping θ restricted to $\mathcal{A}^+(u)$*

A rotor graph such that $S_0 = \emptyset$ is said to be *sinkless*.

We consider graphs that are rotor graphs with $G = (V_0, S_0, \mathcal{A}, h, t, \theta)$, where S_0 may be empty. Unless stated otherwise, all the graphs we refer to follow this structure. As we only deal with rotor graphs in this document, we use the same notation as for multigraphs.

Definition 2.1.10 (Rotor Configuration)**.** *Let $G = (V_0, S_0, \mathcal{A}, h, t, \theta)$ be a rotor graph. A* rotor configuration *of G is a function $\rho : V_0 \to \mathcal{A}$ that assigns to each vertex $u \in V_0$ an outgoing arc $\rho(u) \in \mathcal{A}^+(u)$.*

The set of all rotor configurations on G is denoted by $\mathcal{R}(G)$.

Remark (Rotor configuration representation)**.** *Observe that, for the rest of this document, rotor configurations will be depicted in Figures by arcs in red.*

Definition 2.1.11 (Graph Induced by a Rotor Configuration)**.**
Let $G = (V_0, S_0, \mathcal{A}, h, t, \theta)$ be a rotor multigraph and $\rho \in \mathcal{R}(G)$ be a rotor configuration. The graph induced *by ρ on G, denoted by $G(\rho)$, is a multigraph with the same set of vertices as G, but containing only the arcs in $\rho(V_0)$ (i.e., the set of arcs that are mapped to by ρ). We say that a circuit C is* in *ρ if it belongs to $G(\rho)$.*

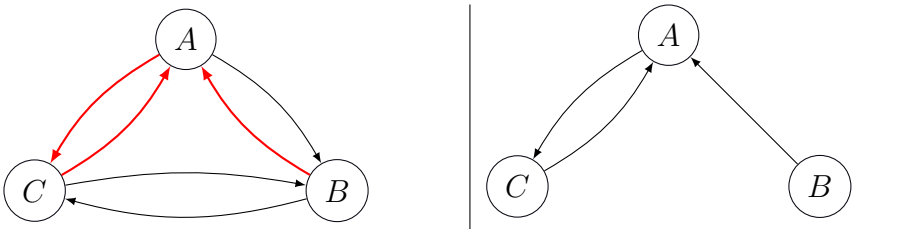
Remark. *In the rotor graph $G(\rho)$, where ρ is a configuration of the rotors, we observe that all non-sink vertices have an outdegree of exactly one. This means that each vertex can belong to at most one cycle in $G(\rho)$.*

Previous remark will be relevant when we will characterize cycle push operations.

Definition 2.1.12 allows us to keep track of the position of the different particles on the graph.

Definition 2.1.12 (Particle Configuration)**.** *A* particle configuration *of a rotor graph G is a mapping σ from V to $\mathbb{Z}$. The set of all particle configurations on rotor graph G is denoted by $\mathcal{P}(G)$.*

We interpret $\sigma(v)$ as the number of particles on v for all $v \in V$.



(a) Rotor configuration ρ depicted in red on a graph G.

(b) Graph induced by ρ denoted by $G(\rho)$

Figure 2.6: Exemple of a graph induced by a rotor configuration.

Observe that if some $v \in V$ exists, such that $\sigma(v) < 0$, vertex v has a deficit of particles. In a context closely related to the *Dollar Game* (see Chapter 1), particles may be interpreted as units of money and so vertices such that $\sigma(v) < 0$ are vertices in debt.

We denote by $0|_V$ the particle configuration such that $0|_V(v) = 0$ for all vertices $v \in V$.

Remark (Particle configurations representation)**.** *Note that for the remaining of this document, particle configurations will be depicted on Figures by numbers on the different vertices.*

Definition 2.1.13 (Degree of Particle Configuration)**.** *Let G be a rotor graph and σ be a particle configuration on G. We define the* degree *of σ denoted $|\sigma|$ as the sum of the number of particles on all vertices of V: $|\sigma| = \sum_{v \in V} \sigma(v)$.*

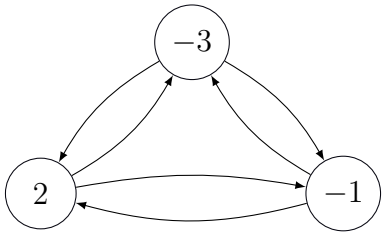


Figure 2.7: The particle configuration represented on this graph has degree -2.

Definition 2.1.14 (Rotor-particle configuration)**.** *Let G be a rotor graph. A* rotor-particle configuration *on G is a pair (ρ, σ) where ρ is a rotor configuration on G and σ is a particle configuration on G. We denote by $\mathcal{RP}(G)$ the set of all rotor particle configurations on G, which is equal to the Cartesian product $\mathcal{R}(G) \times \mathcal{P}(G)$.*

In other words, a rotor-particle configuration specifies the position on the graph of both the rotors and the particles.

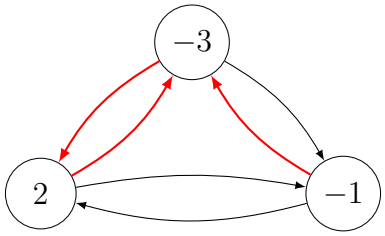


Figure 2.8: Rotor particle configuration on a graph with the rotor configuration depicted by the arcs drawn in red, and the particle configuration depicted by the numbers written on vertices.

2.1.1.2 Routing Operators

Let us define two mappings on $\mathcal{R}(G)$ with $v \in V_0$:

Definition 2.1.15 (Turn operators)**.** *The mapping* $\mathbf{turn}_v^+ : \mathcal{R}(G) \longrightarrow \mathcal{R}(G)$ *is defined by*

$$\mathbf{turn}_v^+(\rho) = \rho'$$

where ρ' *is equal to* ρ *except for* v *where* $\rho'(v) = \theta_v(\rho(v))$*. We naturally denote by* $\mathbf{turn}_v^-$ *the inverse operator:*

$$\mathbf{turn}_v^- : \mathcal{R}(G) \longrightarrow \mathcal{R}(G)$$

defined by

$$\mathbf{turn}_v^-(\rho) = \rho'$$

where ρ' *is equal to* ρ *except for* v *where* $\rho'(v) = \theta_v^{-1}(\rho(v))$*.*

As the $\mathbf{turn}$ operations commute, given W a subset of V_0, we define by $\mathbf{turn}_W^k(\rho)$ the action of applying operation $\mathbf{turn}_v^k(\rho)$ for all $v \in W$ with $k \in \{+, -\}$. Note that in order to simplify the notations we denote by ρ^+ (resp., ρ^-) the rotor configuration $\mathbf{turn}_{V_0}^+(\rho)$ (resp., $\mathbf{turn}_{V_0}^-(\rho)$).

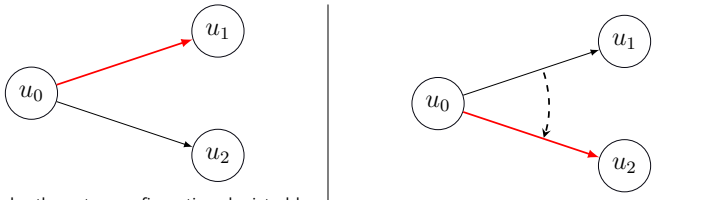
To simplify notation, we introduce an operation that applies the turn operation to a rotor-particle configuration. For a given rotor-particle configuration (ρ, σ) and a vertex v, we define:

$$\mathbf{turn}_v^+(\rho, \sigma) = (\mathbf{turn}_v^+(\rho), \sigma)$$

$$\mathbf{turn}_v^-(\rho, \sigma) = (\mathbf{turn}_v^-(\rho), \sigma)$$

In other words, applying $\mathbf{turn}_v^+$ (resp., $\mathbf{turn}_v^-$) to a rotor-particle configuration amounts to applying $\mathbf{turn}_v^+$ (resp., $\mathbf{turn}_v^-$) to the rotor configuration while leaving the particle configuration unchanged. Figure 2.9b shows an example of a *turn* operation.

Let us define two mappings on $\mathcal{RP}(G)$ where v is a vertex of V_0:



(a) Let ρ be the rotor configuration depicted by the red arcs. The cyclic sequence of arcs obtained from θ_{u_0} is $\{(u_0, u_1), (u_0, u_2)\}$.

(b) The red rotor configuration is obtained by processing the operation: $\mathbf{turn}^+_{u_0}(\rho)$.

Figure 2.9: Example of a $\mathbf{turn}$ operation.

Definition 2.1.16 (Move Operators)**.** *The mapping* $\mathbf{move}_v^+ : \mathcal{RP}(G) \longrightarrow \mathcal{RP}(G)$ *is defined by*

$$\mathbf{move}_v^+(\rho, \sigma) = (\rho, \sigma')$$

where σ' *is defined as follows. We have* $\sigma'(v) = \sigma(v) - 1$ *and* $\sigma'(h(\rho(v))) = \sigma(h(\rho(v)) + 1$*. For all other vertices* $w \in V \setminus \{v, h(\rho(v))\}$*, we have* $\sigma'(w) = \sigma(w)$*. We naturally define the inverse operation* $\mathbf{move}_v^- : \mathcal{RP}(G) \longrightarrow \mathcal{RP}(G)$ *by*

$$\mathbf{move}_v^-(\rho, \sigma) = (\rho, \sigma')$$

This mapping is defined in the same way as $\mathbf{move}_v^+$*, but with the roles of* v *and* $h(\rho(v))$ *reversed. Specifically,* $\sigma'(v) = \sigma(v) + 1$ *and* $\sigma'(h(\rho(v))) = \sigma(h(\rho(v))) - 1$*. For all other vertices* $w \in V \setminus \{v, h(\rho(v))\}$*, we have* $\sigma'(w) = \sigma(w)$*.*

Remark. *In the previous definition, for any* $v \in V_0$*, the operation* $\mathbf{move}_v^+$ *leads to an increase in the value of* $\sigma(h(\rho(v)))$ *and a decrease in the value of* $\sigma(v)$*. Hence, we can interpret it as the movement of a particle departing from* v*, following the arc* $\rho(v)$*, and arriving at vertex* $h(\rho(v))$*. Similarly, the* $\mathbf{move}_v^-$ *operation decreases the value of* $\sigma(h(\rho(v)))$ *and increases the value of* $\sigma(v)$*, suggesting the movement of a particle leaving vertex* $h(\rho(v))$ *and landing on vertex* v*, travelling backwards through the arc* $\rho(v)$*. By drawing an analogy with physics and the concepts of matter and antimatter, we can envision that an antiparticle is departing from vertex* v *along the arc* $\rho(v)$ *and arriving at vertex* $h(\rho(v))$*.*

As the $\mathbf{move}$ operations commute, given X a subset of V_0, we define by $\mathbf{move}_X^k(\rho, \sigma)$ the action of applying operation $\mathbf{turn}_v^k(\rho, \sigma)$ for all $v \in X$ with $k \in \{+, -\}$.

By combining these mappings, we can now define the process of *particle routing*, which represents a single step of a rotor routing sequence that we will define later.

Definition 2.1.17 (Positive routing of a particle)**.** *Let* (ρ, σ) *be a rotor particle configuration on* G*. The* positive routing *of a particle on* $v \in V_0$ *is a mapping:*

$$\mathbf{routing}_v^+ : \mathcal{RP}(G) \longrightarrow \mathcal{RP}(G)$$

defined by

$$\mathbf{routing}_v^+(\rho,\sigma) = \mathbf{turn}_v^+(\mathbf{move}_v^+(\rho,\sigma))$$

where $\mathbf{turn}_v^+$ *and* $\mathbf{move}_v^+$ *are the turn and move operations defined previously.*

This operation can be viewed as the movement of a particle first traveling through arc $\rho(v)$, and then $\rho(v)$ is replaced by $\theta_v(\rho(v))$. We say that v is positively routed if we process a positive routing operation on v. This operation is illustrated in Figure 2.10.

Routing Rule

In this document, we use the convention of first moving particles and then turning rotors when defining routing operations. In the move and then turn version, the rotor configuration indicates the upcoming movements of the particles as they move towards the current rotor configuration and it is updated after their travel. For example, given a rotor particle configuration (ρ,σ), if there is a directed path in $G(\rho)$ from a vertex u to a vertex v, then by processing one single positive routing on each vertex of the path, we obtain a particle configuration σ' such that $\sigma'(v) = \sigma(v)+1$, $\sigma'(u) = \sigma(u)-1$, and for all vertices $w \in V_0 \setminus \{u,v\}$, we have $\sigma'(w) = \sigma(w)$. This can be seen as a single particle traveling from u to v. Therefore, the initial rotor configuration ρ indicates the path the particle is going to follow when vertices are routed up to the point where the path cycles.

The rule where we move first and then turn is commonly used in the study of the ARRIVAL reachability problem, which we will introduce in the next chapter. It has also been utilized in the literature, such as [31] and [40].

In contrast, for the routing rule where the turning operation is performed before the movement operation, the rotor configuration reflects the previous movements of the particles. Given a rotor particle configuration (ρ,σ), suppose there are vertices v and w with a directed path between them in $G(\rho^+)$. Then, by processing a single routing with this rule on each of these vertices, we obtain the rotor particle configuration (ρ',σ') where there is a directed path between v and w in $G(\rho')$, and such that $\sigma'(w) = \sigma(w)+1$, $\sigma'(v) = \sigma(v)-1$, and $\sigma' = \sigma$ for the rest of the vertices. This can once again be seen as a single particle traveling from v to w. However, in this case, it is ρ' that indicates the path the particle has gone through to reach vertex w.

The rule where we turn first and then move is often used in the investigation of the algebraic properties of rotor-routing and the connections between rotor-routing and Markov chains. Several previous studies have employed this rule, including [81], [80], [50], and [41].

One can say that the rule where we move first and then turn represents the future of the particle and the rule where we turn first and then move represents the past of the particle.

Nevertheless, each positive routing with the rule where we move first and then turn can be simulated by a positive routing where turn is applied before move up to applying a $\mathbf{turn}^-_{V_0}(\rho)$ operation beforehand and then applying a $\mathbf{turn}^+_{V_0}(\rho')$ at the end of the routing. Symmetrically, we obtain the reverse implication which proves that these two rules are equivalent.

Definition 2.1.18 (Negative routing of a particle)**.** *Let (ρ, σ) be a rotor-particle configuration on G. The* negative routing *of a particle on $v \in V_0$ is the reverse mapping of a positive routing:*

$$\mathbf{routing}^-_v : \mathcal{RP}(G) \longrightarrow \mathcal{RP}(G)$$

such that

$$\mathbf{routing}^-_v(\rho, \sigma) = \mathbf{move}^-_v(\mathbf{turn}^-_v(\rho, \sigma))$$

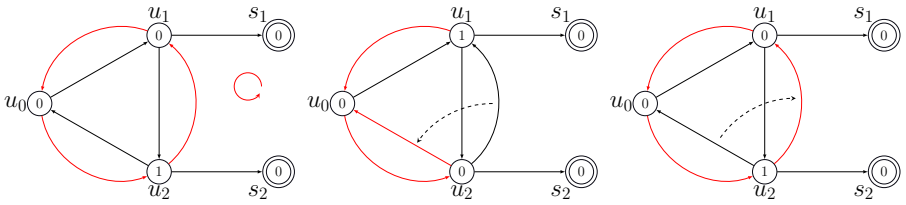
where $\mathbf{turn}^-_v$ and $\mathbf{move}^-_v$ are the turn and move operations defined previously.

Our definition of negative routing ensures that, starting from the rotor-particle configuration (ρ, σ), the following identity holds:

$$\mathbf{routing}^+_v(\mathbf{routing}^-_v(\rho, \sigma)) = \mathbf{routing}^-_v(\mathbf{routing}^+_v(\rho, \sigma)) = (\rho, \sigma).$$

This operation can be represented as replacing $\rho(v)$ with $\theta_v^{-1}(\rho(v))$, followed by the particle moving backwards (and the antiparticle moving forward) from $h(\rho'(v))$ to v. We say that v is negatively routed if we process a negative routing operation on v. This operation is illustrated in Figure 2.10.

Similarly to the positive routing case, a sequence of negative routings along a directed path of $G(\rho^-)$ can be interpreted as the trajectory of an antiparticle moving along this path, given a rotor configuration ρ.



(a) Rotor particle configuration (ρ, σ).

(b) Rotor particle configuration (ρ', σ') obtained from operation $\mathbf{routing}^+_{u_2}(\rho, \sigma)$.

(c) Rotor particle configuration (ρ, σ) obtained from operation $\mathbf{routing}^-_{u_2}(\rho', \sigma')$.

Figure 2.10: The sink-vertices are s_1 and s_2. The red arcs represent the current rotor configuration. The rotor orders on the different vertices are anticlockwise, i.e., the sequence obtained from θ_{u_0} is {(u_0,u_2),(u_0,u_1)}, from θ_{u_1} is {(u_1,u_0),(u_1,u_2),(u_1,s_1)} and from θ_{u_2} is {(u_2,u_1),(u_2,u_0),(u_2,s_2)}.

Definition 2.1.19 (Rotor Routing Sequence)**.** *A* rotor routing sequence *is a (finite or infinite) sequence of rotor particle configurations* $(\rho_i, \sigma_i) \in \mathcal{RP}(G)$ *such that for any* $i \geq 0$*, there exists a sign* s_i *(- or +) and a vertex* $v_i \in V$ *satisfying* $(\rho_{i+1}, \sigma_{i+1}) = \mathbf{routing}_{v_i}^{s_i}(\rho_i, \sigma_i)$*.*

This definition allows the presence of both positive and negative routings in the same routing sequence.

We say that r is a routing sequence from (ρ, σ) to (ρ', σ') if r is a finite sequence and its first and last elements are (ρ, σ) and (ρ', σ'), respectively. We also use the terms *positive routing sequence* and *negative routing sequence* to describe a routing sequence that consists entirely of positive or negative routings, respectively. Following lemma is a generalization of Lemma 3.9 in [50] to routing sequences with both positive and negative routing operations.

Lemma 2.1.20 (Commutativity of routing operations)**.** *All positive and negative routing operations commute.*

Proof. When considering the operators $\mathbf{routing}_{v_1}^{s_1}$ and $\mathbf{routing}_{v_2}^{s_2}$, we can see that for vertices that are not in $\{v_1, v_2\}$, these operators act on different vertices, so they clearly commute. However, for vertices that are in $\{v_1, v_2\}$, these operators either have the same sign or are inverse operators, and thus they also commute. □

Thus, by commutativity, given a subset of vertices $X \subset V_0$ and a rotor particle configuration (ρ, σ), we define operation $\mathbf{routing}_X^s(\rho, \sigma)$ as the action of proceeding $\mathbf{routing}_v^s(\rho, \sigma)$ for all $v \in X$ and s being the sign of the routing.

Commutativity of routing operations is illustrated in Figure 2.11.

In the case of strongly connected graphs, not only routing operations commute but they can be simulated by one another.

Lemma 2.1.21 (Negative routings can be simulated by positive routings and vice versa)**.** *Given a graph* G *such that* $S_0 = \emptyset$ *and* G *is strongly connected, given two rotor particle configurations* (ρ, σ) *and* (ρ', σ') *on* G *such that there is a routing sequence from* (ρ, σ) *to* (ρ', σ')*, there exists a positive routing sequence (a routing sequence with only positive routings) from* (ρ, σ) *to* (ρ', σ') *and a negative routing sequence (a routing sequence with only negative routings) from* (ρ, σ) *to* (ρ', σ')*.*

Proof. Assume that there exists a routing sequence r from (ρ, σ) to (ρ', σ').

Then, it suffices to show that any negative routing in r can be achieved by performing multiple positive routings, and any positive routing in r can be achieved by performing multiple negative routings.

We can specify a negative routing at vertex v in r that transforms the rotor particle configuration (ρ_i, σ_i) into rotor particle configuration $(\rho_{i+1}, \sigma_{i+1})$ by the following:

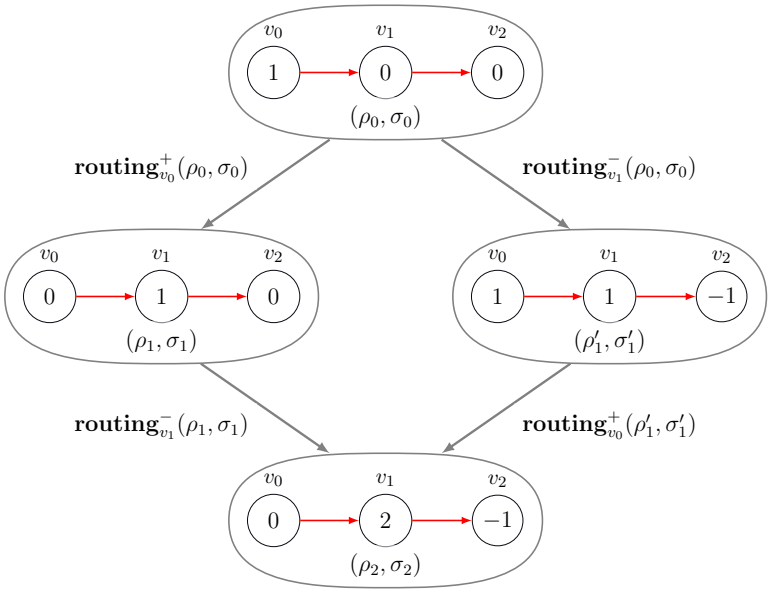


Figure 2.11: Commutativity of routing operations. Changing the order of $\mathbf{routing}^+_{v_0}$ and $\mathbf{routing}^-_{v_1}$ in the routing sequence does not change the resulting rotor particle configuration.

- $\sigma_{i+1}(v) = \sigma_i(v) + 1$
- $\rho_{i+1}(v) = \theta^-_v(\rho_i(v))$
- $\sigma_{i+1}(\rho_{i+1}(v)) = \sigma_i(\rho_{i+1}(v)) - 1$

We construct a positive routing sequence from (ρ_i, σ_i) to $(\rho_{i+1}, \sigma_{i+1})$ as follows. First we proceed exactly $p(w) \cdot |\mathcal{A}^+(w)|$ positive routings on all vertices w except v, with p being the reduced period vector of G. Then we proceed $p(v) \cdot |\mathcal{A}^+(v)| - 1$ positive routings at v. After this sequence of positive routings, we have $\rho_i(w) = \rho_{i+1}(w)$ and $\sigma_{i+1}(w) = \sigma_i(w)$ for all $w \neq v$, and $\rho_{i+1}(v) = \theta^-_v(\rho_i(v))$ and $\sigma_{i+1}(v) = \sigma_i(v) + 1$. Therefore, we have constructed a positive routing sequence from (ρ_i, σ_i) to $(\rho_{i+1}, \sigma_{i+1})$ which shows that any negative routing at v can be achieved by performing multiple negative routings.

The case of a positive routing is similar, we construct the exact same sequence that for the negative case but where all routings are negative instead of positive. □

Rotor Particle Equivalence

Definition 2.1.22 (Rotor Particle Equivalence)**.** *Two rotor particle configurations* (ρ, σ) *and* (ρ', σ') *are said to be* equivalent *if there exists a routing sequence from* (ρ, σ) *to* (ρ', σ')*. We note* $(\rho, \sigma) \sim (\rho', \sigma')$*.*

Definition 2.1.22 satisfies reflexivity, symmetry, and transitivity properties. Therefore, the relation $\sim$ is an equivalence relation on $\mathcal{RP}(G)$.

We denote the fact that r is a routing sequence from (ρ, σ) to (ρ', σ') by $(\rho, \sigma) \overset{r}{\sim} (\rho', \sigma')$.

In the next chapter, our primary focus will be the study of reachability problems. However, it is important to mention that determining whether two rotor particle configurations are equivalent is commonly referred to as *linear equivalence*, drawing an analogy with this relation. The term *linear equivalence* was introduced in [80].

2.2 . Routing Vector

In this subsection, we present the concept of a *routing vector*, which serves as an alternative representation of a routing sequence. The routing vector offers an alternative to the *flow*, which is a commonly used concept in the literature (e.g., [31, 40, 77]), particularly for defining the complexity class associated with the ARRIVAL problem (which we will discuss in Section 3).

Definition 2.2.1 (Routing vector (introduced in [81])). *A* routing vector R *defined on* V *is a function from* V *to* $\mathbb{Z}$.

A routing sequence r from a rotor particle configuration (ρ, σ) to (ρ', σ') naturally gives rise to a *routing vector* R associated with r. The routing vector R is a mapping from each vertex to the difference between the number of positive routings and the number of negative routings it undergoes in r. This routing vector is also referred to as the *induced routing vector* of r. As an example, given a routing sequence r from (ρ, σ) to some (ρ', σ') and given that the sequence of vertices of V routed in r is $v_0^+, v_1^+, v_2^-, v_2^-, v_1^-, v_0^+$, with "$+$" corresponding to a positive routing and "$-$" to a negative routing. Since the routing vector associates to each vertex the difference between the number of positive and negative routings, then the induced routing vector R of r is such that $R(v_0) = 2, R(v_1) = 0, R(v_2) = -2$ and for any other vertex v_i of V, $R(v_i) = 0$.

According to Lemma 2.1.20, there can be multiple routing sequences that correspond to a single routing vector. However, despite the different routing sequences, they all result in the same rotor particle configuration. This observation is summarized in the following lemma.

Lemma 2.2.2. *Given a particle configuration* (ρ, σ) *and given two routing sequences* $(\rho, \sigma) \overset{r_1}{\sim} (\rho_1, \sigma_1)$ *and* $(\rho, \sigma) \overset{r_2}{\sim} (\rho_2, \sigma_2)$ *such that* r_1 *and* r_2 *have the same induced routing vector, then we have* $(\rho_1, \sigma_1) = (\rho_2, \sigma_2)$.

We give an example of computation of the ending rotor particle configuration given that the routing sequence applied has routing vector R to illustrate this lemma.

Let R be the induced routing vector of r_1 and r_2. The rotor particle configuration obtained by applying R to (ρ, σ) is unique. Consider a vertex $v \in V$. Let $\mathcal{S}(R_1)(v)$ be the sequence of vertices such that if $R(v) > 0$ (resp., $R(v) < 0$), $\mathcal{S}(R_1)(v) = h(\rho(v)), h(\theta_v(\rho(v)), h(\theta_v^2(\rho(v)), \ldots, \theta_v^{R(v)-1}(\rho(v))$ (resp., $\mathcal{S}(R_1)(v) = h(\rho(v)), h(\theta_v^{-1}(\rho(v)), h(\theta_v^{-2}(\rho(v)), \ldots, \theta_v^{-R(v)+1}(\rho(v)))$. This is the sequence of vertices that appears at the head of $\rho(v)$ while proceeding $R(v)$ routings at v. Let $|\mathcal{S}(R_1)(v)|_w$ be the number of times vertex w appears in $\mathcal{S}(R_1)(v)$. The rotor configuration $\rho_{\{v\}}$ obtained after proceeding $R(v)$ positive (resp., negative if $R(v) < 0$) routings at v is such that $\rho_{\{v\}}(v) = \theta_v^{R(v)-1}(\rho(v))$ and for any other vertex the rotor configuration remains unchanged. The particle configuration $\sigma_{\{v\}}$ obtained after proceedings those routings is such that $\sigma_{\{v\}} = \sigma(v) - R_1(v)$, for all $w \in \Gamma^-(v)$ if $R(v) > 0$, we have $\sigma_{\{v\}}(w) = \sigma(w) + |\mathcal{S}(R_1)(v)|_w$ and for any other vertex, the particle configuration remains unchanged. Next, repeat the process for all vertices such that $R(v) \neq 0$ to obtain the configuration (ρ_1, σ_1).

We say that R is a routing vector from (ρ, σ) to (ρ_1, σ_1). We denote the fact that R is a routing vector from (ρ, σ) to (ρ_1, σ_1) by $(\rho, \sigma) \overset{R}{\sim} (\rho_1, \sigma_1)$.

Given two rotor particle configurations $(\rho, \sigma), (\rho', \sigma')$, the question of determining whether there exists a routing vector R such that $(\rho, \sigma) \overset{R}{\sim} (\rho', \sigma')$ will be adressed later in this document. This is what is called Linear Equivalence in [80].

2.2.1 . Reduced routing sequence and routing vector

Given two rotor particle configurations (ρ, σ) and (ρ', σ'), we are aware that there might exist an infinite number of routing sequences r such that $(\rho, \sigma) \overset{r}{\sim} (\rho', \sigma')$. However, for convenience, it is preferable to focus on routing sequences of the following form.

Definition 2.2.3 (Reduced routing sequence)**.** *A* reduced routing sequence*, associated with a routing vector R, is a routing sequence where each routed vertex is either only positively routed or only negatively routed, and the induced routing vector of the sequence is equal to R.*

Remark. *Given a reduced routing sequence r, the routing vector of r is exactly the number of times each vertex is routed in r, signed negatively if the routings are negative.*

Routing vectors from (ρ, σ) to (ρ, σ) can be represented in the form $k_v \cdot |\mathcal{A}^+(v)|$ for all $v \in V$, where k_v is an integer. It is worth noting that for any given period vector of a graph G, denoted by p, we have $p \cdot L_G = 0$. Consequently, $\sigma = \sigma + p \cdot L_G$, which implies that by routing each vertex v exactly $p(v) \cdot |\mathcal{A}^+(v)|$ times, we obtain a routing vector from (ρ, σ) to (ρ, σ). This is illustrated in Figure 2.12. Therefore, a period vector induces a routing vector that does not alter the rotor particle configuration, thus justifying the term *period vector*.

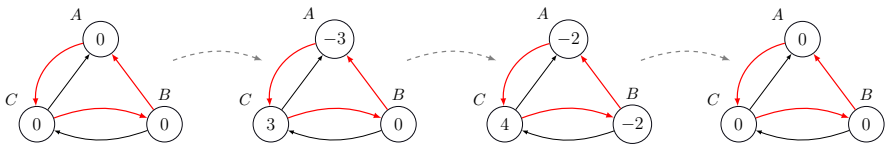


Figure 2.12: Example of a routing vector induced by a period vector. The vector $p = \begin{pmatrix} 3 & 1 & 2 \end{pmatrix}$ is a period vector of this graph. Let (ρ, σ) be the rotor particle configuration depicted on the left. Let $R_p = (A : 3, B : 2 : C : 4)$ obtained by computing $p(v) \cdot |\mathcal{A}^+(v)|$ for each vertex $v \in V$. Then $(\rho, \sigma) \overset{R_p}{\sim} (\rho, \sigma)$.

Let R be a routing vector where $R(v) = |\mathcal{A}^+(v)|$ and for all other vertices w, we have $R(w) = 0$. It is worth mentioning that given two particle configurations σ and σ', the two following propositions are equivalent:

- For all rotor configuration ρ, $(\rho, \sigma) \overset{R}{\sim} (\rho, \sigma')$
- $\sigma' = \sigma + 1_v \cdot L_G$

where 1_v is a vector of size $|V|$ with a value of 1 for vertex v and a value of 0 for any other vertex in V.

Proposition 2.2.4. *Let G be a stopping rotor graph. Given two rotor particle configurations (ρ, σ) and (ρ', σ') on G, if two routing vectors R_1 and R_2 are such that $(\rho, \sigma) \overset{R_1}{\sim} (\rho', \sigma')$ and $(\rho, \sigma) \overset{R_2}{\sim} (\rho', \sigma')$, then $R_1 = R_2$.*

Proof. Since the only possible period vector on a stopping graph is $0|_{|V|}$, then, any routing vector from (ρ, σ) to itself equals $0|_{|V|}$, thus, $R_1 - R_2 = 0|_{|V|}$ which implies that $R_1 = R_2$. □

However, in the case where the graph is strongly connected, it is easy to see that there may exist two rotor particle configurations (ρ, σ) and (ρ', σ') such that there is an infinite number of routing vectors from (ρ, σ) to (ρ', σ') (see Figure 2.13).

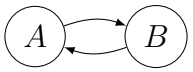


Figure 2.13: Let (ρ, σ) be any rotor particle configuration on this graph with two vertices, any routing vector R of the form $(A : k, B : k)$ with $k \in \mathbb{Z}$ is a routing vector from (ρ, σ) to itself.

Now we define one particular routing vector that is representative of the routing sequences between two rotor particle configurations.

Definition 2.2.5 (Reduced Routing Vector)**.** *Let R be a routing vector between two rotor particle configurations on G. The routing vector R is considered* reduced

if for any strongly connected component (sink component) C_i of G with primitive period vector p_{C_i}, $R(C_i) \geq 0$ and for all vertices $v \in C_i$ we have $R(v) \leq p_{C_i}(v) \cdot |\mathcal{A}^+(v)|$.

Definition 2.2.5 implies that on a stopping rotor graph, any positive routing vector is reduced since sinks cannot be routed and the only period vector of a sink component containing only a single sink is 0.

Remark. *Since any graph can be decomposed in strongly connected components (sink components) and stopping graphs (transient components), hence, it suffices to show the existence and unicity of reduced routing vector of strongly connected graphs and stopping graphs to prove the existence and unicity of the reduced routing vector of a general digraph.*

Proposition 2.2.6 (Existence and Unicity of Reduced Routing Vector)**.** *Let (ρ, σ) and (ρ', σ') be two rotor particle configurations.*

- *On a strongly connected graph, if $(\rho, \sigma) \sim (\rho', \sigma')$, then there exists a reduced routing vector $\hat{R}$ such that $(\rho, \sigma) \overset{\hat{R}}{\sim} (\rho', \sigma')$*
- *On a stopping graph if $(\rho, \sigma) \overset{R}{\sim} (\rho', \sigma')$ with $R \geq 0$, then R is reduced*
- *If R_1 and R_2 are two reduced routing vectors from (ρ, σ) to (ρ', σ'), then $R_1 = R_2$*

Proof. In the case where G is a stopping graph, Proposition 2.2.4 states that if $(\rho, \sigma) \sim (\rho', \sigma')$, then there exists a unique routing vector from (ρ, σ) to (ρ', σ') which proves unicity when combined with our previous remark on stopping graphs.

If G is strongly connected and $(\rho, \sigma) \sim (\rho', \sigma')$, Lemma 2.1.21 tells us that there exists a positive routing vector R such that $(\rho, \sigma) \overset{R}{\sim} (\rho', \sigma')$. In this case, two scenarios are possible.

First, if R is reduced, then we already have a reduced routing vector from (ρ, σ) to (ρ', σ').

Second, if there exists a positive period vector p (there always exists one since the primitive period vector of a strongly connected graph is strictly positive) such that for all v, $R(v) > p(v) \cdot |\mathcal{A}^+(v)|$, we can define a new routing vector R' with values $R'(v) = R(v) - p(v) \cdot |\mathcal{A}^+(v)|$. This new routing vector R' satisfies $(\rho, \sigma) \overset{R'}{\sim} (\rho', \sigma')$, and we can repeat the process by considering R' instead of R. This process can be repeated iteratively until we obtain a reduced routing vector from (ρ, σ) to (ρ', σ').

In the general case, assume that R_1 and R_2 are two reduced routing vector from (ρ, σ) to (ρ', σ'). Since the vector $(R_1(v) - R_2(v))$ is a routing vector from (ρ, σ) to itself, then $(R_1(v) - R_2(v))/|\mathcal{A}^+(v)|$ is a period vector t of G.

From Proposition 2.1.5, we have that $t = k * p$ with $k \in \mathbb{Z}$ and p being the primitive period vector of G. Since p is positive, we have that t is either all positive or all negative, which means that either $R_1(v) \geq R_2(v)$ for all $v \in V$ or $R_1(v) \leq R_2(v)$ for all $v \in V$.

Thus, either we have that for all $v \in V$, $R_1(v) \geq |t(v)| * |\mathcal{A}^+(v)| + R_2(v)$ or we have that for all $v \in V$, $R_2(v) \geq |t(v)| * |\mathcal{A}^+(v)| + R_1(v)$. This implies that either R_1 or R_2 is not reduced which is a contradiction.

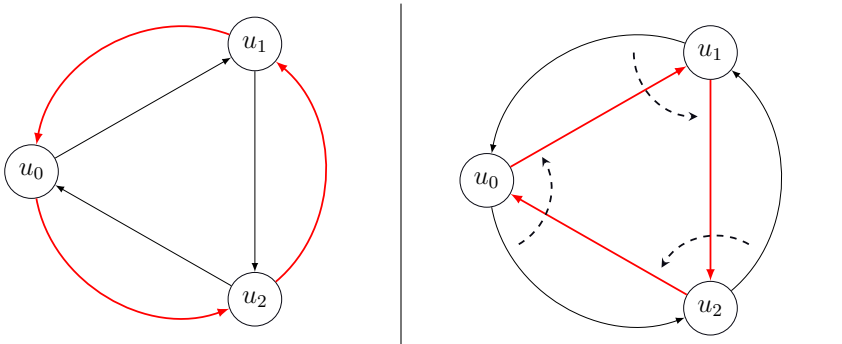
□

2.3 . Cycle Pushes and Equivalence Classes of Rotor Configurations

In this section, we define an existing operation called *cycle push* on $\mathcal{R}(G)$, which helps us create equivalence classes within $\mathcal{R}(G)$ and $\mathcal{P}(G)$. This operation has been introduced to study the orientations of lattices in [73]. It is worth noting that in this context, we refer to a "cycle" as a "circuit" in graph theory, but we will stick with the term *cycle push* to maintain consistency with the existing literature (see [50]).

Definition 2.3.1 (Positive Cycle Push)**.** *Given a circuit C in $G(\rho)$ with ρ being a rotor configuration, a* positive cycle push *on C is the operation $\mathbf{turn}_C^+(\rho)$ which transforms ρ into ρ' such that $\rho'(v_i) = \theta_{v_i}(\rho(v_i))$ for all vertices $v_i \in C$, and $\rho'(v) = \rho(v)$ for all vertices $v \notin C$.*

This operation is illustrated in Figure 2.14.



(a) Rotor configuration ρ.

(b) Rotor configuration ρ' obtained from a positive cycle push operation on the circuit of $G(\rho)$ formed by arcs $\{(u_0,u_2),(u_2,u_1),(u_1,u_0)\}$.

Figure 2.14: Positive cycle push operation on the circuit u_0, u_1, u_2. There is a single rotor order possible on this graph since each vertex has outdegree 2.

The positive cycle push operation can also be understood as a specific routing sequence where a single particle is routed along the vertices of circuit C up to its

initial position. This operation "opens" the previously existing circuit in $G(\rho)$, and is essentially an accelerated version of the routing sequence where each vertex of C is routed once.

In the same manner, we also define the reverse operation.

Definition 2.3.2 (Negative Cycle Push)**.** *Let ρ be a rotor configuration. Consider a circuit C in $G(\rho^-)$ (recall that ρ^- is the rotor configuration obtained by the operation $\mathbf{turn}^-_{V_0}(\rho)$). We define the* negative cycle push *operation on C as the operation $\mathbf{turn}^-_C(\rho)$.*

A negative cycle push transforms ρ into ρ' such that $\rho'(v_i) = \theta^{-1}_{v_i}(\rho(v_i))$ for all $v_i \in C$, and $\rho'(v) = \rho(v)$ for all $v \notin C$.

The negative cycle push operation can also be understood as a specific routing sequence where a single antiparticle is routed along the vertices of circuit C of $G(\rho^-)$ up to its initial position. This operation "recloses" the previously opened circuit in $G(\rho)$.

We say that a vertex v is *cycle pushed* if v belongs to a circuit C for which we apply a cycle push operation (positive or negative).

Note that contrary to a routing operation, a cycle push operation needs some prerequisites as the circuit on which we apply a cycle push operation must belong to $G(\rho)$ or $G(\rho^-)$.

2.3.1 . Rotor configuration equivalence classes

Rotor configuration equivalence classes are defined in [41] by using Sandpiles equivalence classes but we characterize them in the following proposition using only rotor configurations.

Proposition 2.3.3. *Given a rotor configuration $\rho_0 \in \mathcal{R}(G)$ and a particle configuration $\sigma \in \mathcal{P}(G)$, the set $\mathcal{R}_1$ of rotor configurations ρ_1 such that $(\rho_0, \sigma) \sim (\rho_1, \sigma)$ is independent of σ.*

The set $\mathcal{R}_1$ is called the rotor class of ρ_0.

Proof. Suppose that ρ' belongs to the rotor class of ρ for a given particle configuration σ. This means that there exists a routing sequence r from (ρ, σ) to (ρ', σ). Now, we show that ρ' belongs to the rotor class of ρ for a given particle configuration σ'. Let, $\sigma^- = \sigma' - \sigma$. Then, $(\rho, \sigma' - \sigma^-) \overset{r}{\sim} (\rho', \sigma' - \sigma^-)$. So as adding σ^- does not change the existence of such a routing sequence, we have our result. □

To keep the formalism used for rotor particle configurations, if two rotor configurations ρ and ρ' belong to the same rotor class, then for any particle configuration σ, $(\rho, \sigma) \sim (\rho', \sigma)$. Thus, we say that ρ and ρ' are *equivalent* and we note $\rho \sim \rho'$.

Proposition 2.3.3 satisfies the reflexivity, symmetry, and transitivity properties. Therefore, the relation $\sim$ is an equivalence relation on $\mathcal{R}(G)$. Furthermore, it implies the following corollary.

Corollary 2.3.4. *Let $\rho_0 \in \mathcal{R}(G)$ and $\sigma, \sigma' \in \mathcal{P}(G)$. Assume there exist two rotor configurations ρ_1 and ρ_2 such that $(\rho_0, \sigma) \sim (\rho_1, \sigma')$ and $(\rho_0, \sigma) \sim (\rho_2, \sigma')$. Then $\rho_1 \sim \rho_2$.*

Proof. Since $(\rho_0, \sigma) \sim (\rho_1, \sigma')$, $(\rho_0, \sigma) \sim (\rho_2, \sigma')$ and "$\sim$" is an equivalence relation, we have $(\rho_1, \sigma') \sim (\rho_2, \sigma')$ which proves that $\rho_1 \sim \rho_2$. □

Let us give a characterization of this equivalence relation.

Theorem 2.3.5. *Given two rotor configurations ρ and ρ', $\rho \sim \rho'$ if and only if ρ' can be obtained from ρ by a sequence of cycle pushes.*

Proof. First, note that if there exists a cycle push sequence from ρ to ρ' then we have $\rho \sim \rho'$. Indeed, cycle push operations can be obtained by routing particles or antiparticles up to their original positions which means that for any $\sigma \in \mathcal{P}(G)$ there is a routing sequence from (ρ, σ) to (ρ', σ). However, the converse needs to be proven. To do this, we assume that ρ and ρ' belong to the same rotor class, and we aim to show that a cycle push sequence exists from ρ to ρ'.

Given $0|_V \in \mathcal{P}(G)$, let R be a reduced routing vector from $(\rho, 0|_V)$ to $(\rho', 0|_V)$. By the fact that R is reduced, each vertex only does either positive or negative routings. This means that each time a vertex is positively (resp., negatively) routed in R, it has to receive a (resp., give) particle from (resp., to) one of its neighbours because the ending particle configuration is still $0|_{V_0}$. Thus, we define three sets of vertices as follows, $R^+ = \{v \in V_0, R(v) > 0\}$, $R^- = \{v \in V_0, R(v) < 0\}$ and the set of vertices $R_0 = \{v \in V_0, R(v) = 0\}$. Observe first that the total number of particles on R^- cannot decrease when applying R. We also have that a particle cannot be routed from a vertex of R^+ to a vertex of R^- when applying R because if it was, the total number of particles on vertices of R^- would be more than zero at the end of the routing which contradicts the fact that the ending configuration is $(\rho', 0|_{V_0})$. This gives us that, if $R^+ \neq \emptyset$, the graph $G^+ = (R^+, \rho(R^+))$ is well defined because for all $v \in R^+$ we have $h(\rho(v)) \in R^+$. Thus, as all vertices have outdegree exactly one in G^+, there necessarily is a circuit C in G^+. And we can proceed a cycle push on C. Now, we proceed by induction, remove one positive routing in R for each vertex of C and repeat the process with this new routing vector until R^+ is empty.

Equivalently we can do the same for R^-, and in that case we show that there necessarily is a circuit in $G^- = (R^-, \rho^-(R^-))$, and we proceed inductively afterwards. □

Characterization of rotor classes of a stopping graph

This subsection provides a detailed examination of the structure of rotor classes on stopping rotor graphs. By maximal positive (resp., negative) cycle push sequence,

we mean a sequence long enough such that there are no more positive (resp., negative) cycle push doable when the sequence terminates.

Lemma 2.3.6. *If G is stopping, any maximal positive (or negative) cycle push sequence is finite.*

Proof. Given a rotor configuration ρ on a stopping graph G, all sink neighbours can only be positively cycle pushed a finite number of times before their only outgoing arc in the graph induced by the rotor configuration is directed towards the sink, rendering them unable to be cycle pushed any further. By connectivity, we can conclude that all vertices can only be cycle pushed a finite number of times. The argument is similar for a negative cycle push sequence but we consider the graph $G(\rho^-)$ instead of G. □

We now define the concept of *recurrent rotor configuration* on a stopping rotor graph. This idea has already been introduced in the literature (see [81]), and it refers to rotor configurations $\rho \in \mathcal{R}(G)$ that satisfy the condition $G(\rho^-)$ is acyclic when G is stopping.

Proposition 2.3.7 (Acyclic Configuration)**.** *Given a stopping rotor graph G, each rotor class contains exactly one acyclic configuration ρ and one configuration such that $G(\rho^-)$ is acyclic.*

Proof. Let G^- be the rotor graph with same vertices and arcs than G but with rotor order θ^{-1}. Given a rotor class, by Lemma 2.3.6, there always exists an acyclic rotor configuration within this class since any maximal positive cycle push sequence is finite. Let ρ be a rotor configuration on G. Any negative cycle push on a set of vertices C in G such that C is a circuit in $G(\rho^-)$ can also be seen as a positive cycle push on C in $G^-(\rho^-)$. This is illustrated in Figure 2.15. And, since any maximal positive cycle push sequence is finite, the sequence of positive cycle pushes on G^- necessarily ends, which proves the existence of ρ.

Proof of unicity is presented in Section 2.4 since it is a more convenient framework. □

Note that acyclic configurations are the directed spanning forests on G rooted on the sink vertices. Now, we state the following Theorem to compute the number of rotor classes on a graph G.

Theorem 2.3.8 (Theorem 4.1 in [70])**.** *The number of rotor classes on a stopping rotor graph G is the determinant of the submatrix of the Laplacian matrix of G obtained by removing all rows and columns corresponding to vertices of S_0.*

The diagram of Figure 2.16 summarizes the construction of rotor classes on a stopping graph G.

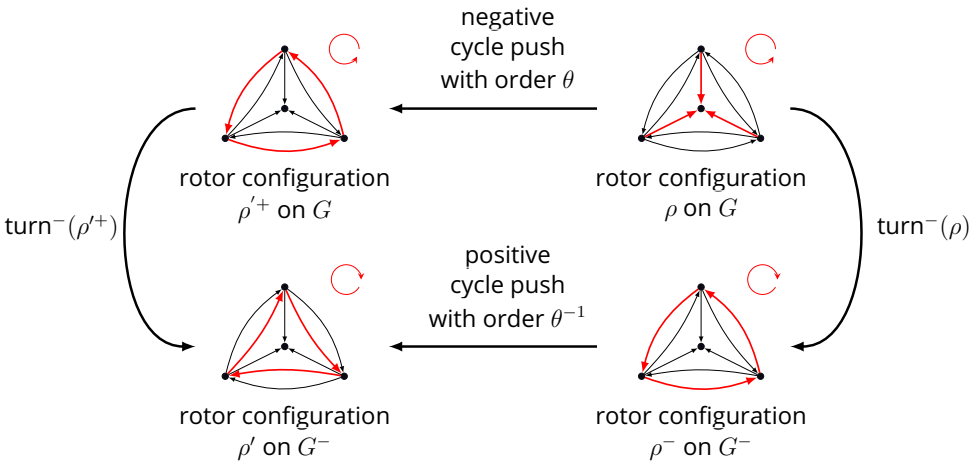


Figure 2.15: Illustration that a negative cycle push on G can be simulated by a positive cycle push on G^-.

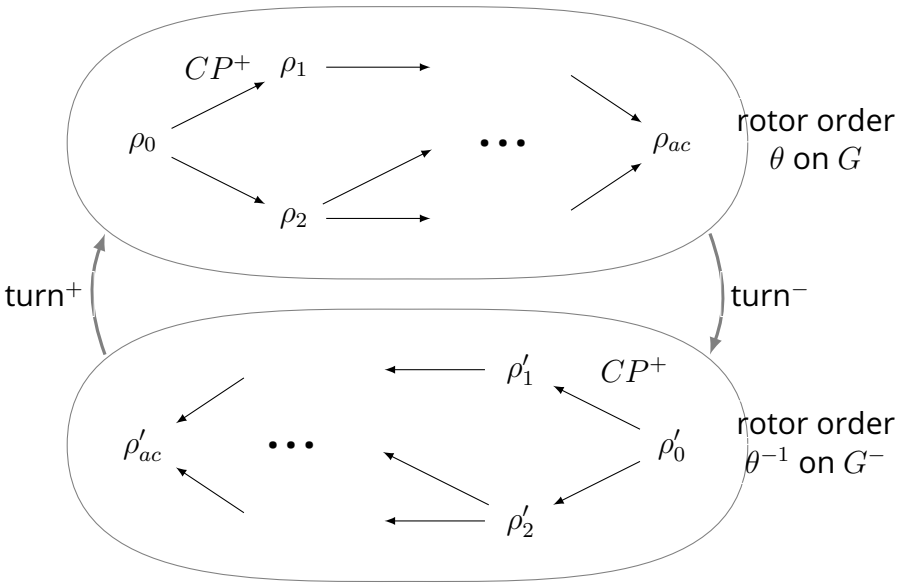


Figure 2.16: Each ρ_i is a rotor configuration on G, each ρ'_i is a rotor configuration on G^-, each arc represents a positive cycle push (denoted by "CP^+"). By Lemma 2.3.6 and Proposition 2.3.7, ρ_{ac} is the acyclic rotor configuration of the rotor class depicted above. Then, by applying a **turn**$^-$ on ρ_{ac}, we obtain the unique configuration that cannot be negatively cycle pushed in G^-. Equivalently, ρ_0 is the only configuration that cannot be negatively cycle pushed in G since ρ'_{ac} is acyclic by Lemma 2.3.6 and Proposition 2.3.7.

2.3.2 . Particle configuration class

Now that we have defined rotor classes, we present some properties on particle configuration classes.

In this subsection, we introduce equivalence classes for particle configurations, which are similar to the ones defined in [12]. In the literature, rotor classes are typically defined in terms of particle configuration equivalence (see [41]), but we approach it differently in this document.

Proposition 2.3.9 (Particle configuration equivalence)**.** *Let $\sigma \in \mathcal{P}(G)$. Given $\rho \in \mathcal{R}(G)$, the set P_1 of particle configuration σ' such that there is a routing sequence from (ρ, σ) to (ρ, σ') is independent of ρ.*

The set P_1 is called the particle class of σ.

Proof. Suppose that σ' belongs to the particle class of σ for a given configuration ρ. This means that there exists a routing sequence r from (ρ, σ) to (ρ, σ'). We want to show that σ' also belongs to the particle class of σ for another configuration ρ'. Since $(\rho, \sigma) \overset{r}{\sim} (\rho, \sigma')$, we have that the routing vector R of r is such that $R(v) = k_v \cdot |\mathcal{A}^+(v)|$ for all $v \in V_0$ with $k_v \geq 0$. In this sequence, each vertex that is routed gives k particles to each of its outgoing neighbours. This does not depend on the initial rotor configuration, which means that $(\rho', \sigma) \overset{r}{\sim} (\rho', \sigma')$.

□

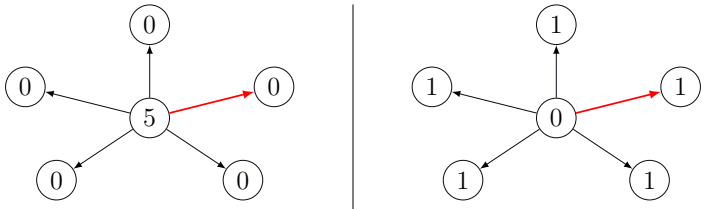
Previous definition satisfies the properties of reflexivity, symmetry, and transitivity. Therefore it is an equivalence relation on $\mathcal{P}(G)$.

Observe that this definition is analogous to the one of rotor classes, with the difference that here we are changing the particle configuration σ instead of the rotor configuration ρ. Thus, we use the notation $\sigma \sim \sigma'$ to denote that σ and σ' are in the same equivalence class, and we say that they are equivalent.

Remark. *Particle configurations within a given equivalence class are exactly the particle configurations that can be obtained from one another by a reduced routing sequence where each vertex appearing in the sequence is routed a number of times equal to a multiple of its outdegree.*

Hence, we naturally define two new operations on $\mathcal{P}(G)$ called positive (resp., negative) *firing* that works as follows:

The action of proceeding a positive (resp., negative) firing on a vertex v consists in performing $|\mathcal{A}^+(v)|$ positive (resp., negative) routings on v. We denote such an operation by $\mathbf{firing}_v^s$ with s being the sign of the firing (- or +). Although this notion already exists for positive routings (e.g. [14]), we extend it to negative routings as well. It should be noted that firing operations do not alter the rotor configuration, as shown in Figure 2.17.



(a) Rotor particle configuration (ρ, σ). (b) Rotor particle configuration (ρ', σ').

Figure 2.17: Rotor particle configuration (ρ', σ') is obtained by proceeding a firing operation on the central vertex from (ρ, σ). One can check that $\rho = \rho'$.

Definition 2.3.10. *A* firing sequence *is a sequence of rotor particle configuration $(\rho_i, \sigma_i) \in \mathcal{RP}(G)$ such that for any $i \geq 0$, there exists a sign s_i (- or +) and a vertex $v_i \in V_0$ satisfying* ***firing*** $\text{firing}_{v_i}^{s_i}(\rho_i, \sigma_i) = (\rho_{i+1}, \sigma_{i+1})$.

Naturally, we define a *firing vector* associated with a firing sequence f from a rotor particle configuration (ρ, σ) to (ρ, σ') by a function from V_0 to $\mathbb{Z}$ that associates with each vertex the number of times it is positively fired in f minus the number of times it is negatively fired in f.

When we have two rotor particle configurations (ρ, σ) and (ρ, σ'), we may wonder if there exists a firing sequence that can take us from (ρ, σ) to (ρ, σ'). Proposition 8 of [52] provides an answer to this question, which we restate in our context.

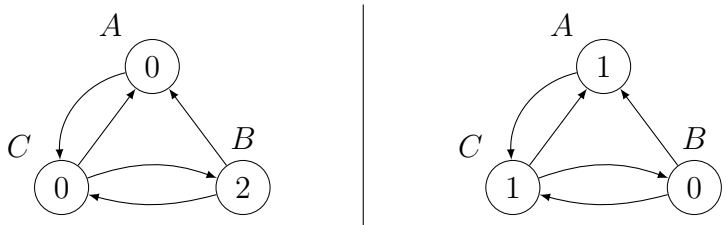
Lemma 2.3.11. *Given a graph G, there exists a polynomial-time algorithm to determine whether there exists a non-negative firing vector from a rotor particle configuration (ρ, σ) to another rotor particle configuration (ρ, σ').*

Note that the algorithm initially computes a firing vector, which may not be non-negative. In the case where the graph is strongly connected, the primitive period vector of G can be added multiple times until the firing vector becomes non-negative. It is possible to compute a firing vector by checking whether $g \cdot L_G = \sigma' - \sigma$ has an integer solution, where L_G is the Laplacian matrix of G. A polynomial time algorithm has been shown to exist for this task, as stated in [44], and it also computes a solution if one exists.

Routing vector is a certificate

This section presents properties of routing vectors that will be utilized in Section 3 as a certification criterion for the complexity of various problems.

We will now prove that for a stopping graph, any routing sequence applied to a rotor particle configuration (ρ, σ) that leads to another rotor particle configuration (ρ', σ') with $\sigma'|_{V_0} = \sigma|_{V_0}$ also satisfies $\sigma'|_{S_0} = \sigma|_{S_0}$.



(a) Particle configuration σ (b) Particle configuration σ'

$$L = \begin{vmatrix} -1 & 0 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{vmatrix} \qquad \begin{aligned} \begin{pmatrix} g_1 & g_2 & g_3 \end{pmatrix} \cdot L &= \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 2 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & -2 & 1 \end{pmatrix} \end{aligned}$$

Figure 2.18: Computation of a non-negative firing vector between two particle configurations σ and σ' on a graph G. Matrix L is the Laplacian matrix of G. By solving the equation in $g \cdot L = \sigma' - \sigma$ in $\mathbb{Z}^3$, we obtain $g = (0, 1, 0)$.

Lemma 2.3.12. *Given a rotor particle configuration (ρ, σ) on a stopping graph G, for any two rotor configurations ρ_1 and ρ_2 such that $(\rho, \sigma) \sim (\rho_1, \sigma'_1)$ and $(\rho, \sigma) \sim (\rho_2, \sigma'_2)$ with $\sigma'_1|_{V_0} = \sigma'_2|_{V_0}$, we have $\sigma'_1 = \sigma'_2$ and more specifically $\sigma'_1|_{S_0} = \sigma'_2|_{S_0}$.*

More precisely, this means that $\sigma'_1|_{S_0}$ does not depend on the choice of ρ'.

Proof. Assume that the graph G only has one sink vertex. It is clear that if $\sigma'_1|_{V_0} = \sigma'_2|_{V_0}$ then $\sigma'_1|_{S_0} = \sigma'_2|_{S_0}$.

If the graph has two or more sinks, this lemma can be restated as the fact that it is impossible to transfer a particle from one sink s_0 to another sink s_1 by a routing sequence that does not modify the particle configuration.

Consider the modified graph G' obtained by merging the vertices s_0 and s_1 into a new sink vertex S. In this modified graph G', any routing vector that transfers a particle from s_0 to s_1 corresponds to a routing from a particle configuration σ' to itself. However, since G' is a stopping graph, the only period vector of G' is the zero vector $0|_{|V|}$. This implies that no particle can be transferred between s_0 and s_1 in any routing sequence, which contradicts the assumption that a particle has been transferred in the given routing sequence. □

2.4 . Positive Rotor Walk

A Positive Rotor walk is the routing sequence obtained by moving a single particle (or antiparticle) along a path as in random walks. It was initially introduced in the literature under this name (e.g., [75]) to draw an analogy with the concept of a random walk. The notion of rotor walk simplifies our definitions and properties in Chapter 3 and is the only type of routing sequence considered in Chapter 4.

2.4.1 . Basic definitions and properties

Definition 2.4.1 (Positive Rotor Walk)**.** *A positive routing sequence* $r = (\rho_0, \sigma_0), (\rho_1, \sigma_1), \ldots, (\rho_k, \sigma_k)$ *is a* positive rotor walk *if for any non negative integer* $i < k-1$*, if* $\mathbf{routing}^+_{v_i}(\rho_i, \sigma_i) = (\rho_{i+1}, \sigma_{i+1})$ *then* $\mathbf{routing}^+_{h(\rho_i(v_i))}(\rho_{i+1}, \sigma_{i+1}) = (\rho_{i+2}, \sigma_{i+2})$*.*

We say that the rotor walk defined in the previous definition is a rotor walk at v_0 since the first routing is at vertex v_0. An example of such a routing sequence is illustrated in Figure 2.19.

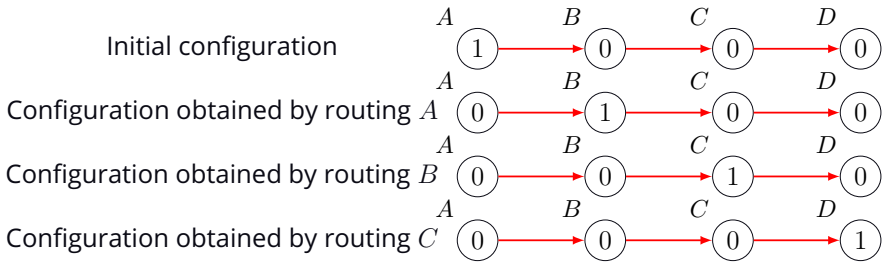


Figure 2.19: Example of a rotor walk at A where the vertices routed are successively A, B, C.

This can be interpreted as the movement of a single particle. Equivalently we define a *negative rotor walk* that is a negative routing sequence with the following property: if $\mathbf{routing}^-_{v_i}(\rho_i, \sigma_i) = (\rho_{i+1}, \sigma_{i+1})$ then $\mathbf{routing}^-_{\rho_{i+1}(v_i)}(\rho_{i+1}, \sigma_{i+1}) = (\rho_{i+2}, \sigma_{i+2})$.

Remark that a routing sequence can always be seen as a union of rotor walks.

Definition 2.4.2 (Path of a rotor walk)**.** *Let* $r = (\rho_0, \sigma_0), (\rho_1, \sigma_1), \ldots, (\rho_k, \sigma_k)$ *be a rotor walk at* v_0*. The* path of the rotor walk r *is the sequence of vertices* $v_0, v_1, \ldots, v_{k-1}$ *routed in* r *to which we add the vertex* v_k *such that* $v_k = \rho_{k-1}(v_{k-1})$ *if* r *is positive and* $v_k = \rho_k(v_{k-1})$ *if* r *is negative.*

A maximal rotor walk is a routing sequence obtained by moving a particle (or antiparticle) until it reaches a sink if it is finite.

Definition 2.4.3 (Maximal Rotor Walk)**.** *A positive (resp., negative) rotor walk* $r = (\rho_0, \sigma_0), (\rho_1, \sigma_1), \ldots, (\rho_k, \sigma_k)$ *with path* $v_0, v_1, \ldots, v_{k-1}$ *on a stopping graph is* maximal *if* $h(\rho_{k-1}(v_{k-1}))$ *(resp.,* $h((\rho_k)(v_{k-1}))$*) is a sink. On a strongly connected graph, a rotor walk is maximal if it is infinite in length.*

Since rotor walk can be seen as the movement of a single particle (or antiparticle), we have the following properties.

Proposition 2.4.4 (Rotor Walk covering)**.** *Given a strongly connected graph G, any maximal rotor walk (positive or negative) is such that all vertices will be routed an infinite number of times in such a rotor walk.*

Proof. Since there are no sinks in the graph, it is possible to perform a positive or negative routing at any vertex. Let I be the set of vertices that are routed infinitely many times. Since the graph is strongly connected, there must exist an arc in G with its tail in I and its head in $V \setminus I$. This arc will be visited an infinite number of times in the rotor walk. This implies that there is at least one vertex in $V \setminus I$ that is visited an infinite number of times, which contradicts the assumption that only vertices in I are visited infinitely often. Therefore, all vertices are routed an infinite number of times in the rotor walk. □

This proposition implies the following corollary.

Corollary 2.4.5. *If G is stopping, any positive (or negative) maximal rotor walk in G is finite.*

Proof. It is easy to see that the particle (or antiparticle) will necessarily reach a sink vertex at some point which will end the rotor walk. □

From the fact that each vertex only has one outgoing arc in $G(\rho)$, we have the following proposition.

Proposition 2.4.6 (Unicity of Maximal rotor walk)**.** *Given two maximal rotor walks r_1 and r_2 at v_0 starting from the same rotor particle configuration (ρ, σ), we have $r_1 = r_2$.*

Now, we state a Lemma that we will use several times in Chapter 3.

Lemma 2.4.7 (Moving particles and Antiparticles)**.** *Given a rotor graph G, and given a rotor particle configuration (ρ, σ),*

1. *If G is strongly connected, for all vertex $v \in V_0$ and $w \in V_0$, there exists a positive (resp., negative) rotor walk at v from (ρ, σ) to some (ρ', σ') such that $\sigma'(w) = \sigma(w) + 1$ (resp., $\sigma'(w) = \sigma(w) - 1$), $\sigma'(v) = \sigma(v) - 1$ (resp., $\sigma'(v) = \sigma(v) + 1$) and for all vertex $u \in V_0 \setminus \{v, w\}$ we have $\sigma'(u) = \sigma(u)$*
2. *If G is stopping, for all vertices $v \in V_0$, there exists a positive (resp., negative) rotor walk at v from (ρ, σ) to some (ρ', σ') such that $\sigma'(v) = \sigma(v) - 1$ (resp., $\sigma'(v) = \sigma(v) + 1$) and for all vertices $u \in V_0 \setminus \{v\}$ we have $\sigma'(u) = \sigma(u)$*

Second implications means that there exists a sink vertex that gains or loses a particle in the process.

Proof. First we prove Implication (1). In the case of a sinkless strongly connected graph, let $u, w \in V_0$, we have by Lemma 2.4.4 that there exists a positive routing sequence from (ρ, σ) to some (ρ', σ') such that $\sigma'(w) = \sigma(w) + 1$, $\sigma'(v) = \sigma(v) - 1$ and for all vertices $u \in V_0 \setminus \{v, w\}$ we have $\sigma'(u) = \sigma(u)$. But, by Lemma 2.1.21, this also shows that there exists a negative routing sequence from (ρ, σ) to some (ρ', σ') such that $\sigma'(w) = \sigma(w) - 1$, $\sigma'(v) = \sigma(v) + 1$ and for all vertices $u \in V_0 \setminus \{v, w\}$ we have $\sigma'(u) = \sigma(u)$.

Now we prove Implication (2). In the case of a stopping graph, by Proposition 2.4.4 we have that there exists a finite maximal positive rotor walk starting from a routing at vertex v. Hence showing that there exists a positive routing sequence from (ρ, σ) to some (ρ', σ') such that $\sigma'(v) = \sigma(v) - 1$ and for all vertices $u \in V_0 \setminus \{v\}$ we have $\sigma'(u) = \sigma(u)$. Equivalently, there exists a finite maximal negative rotor walk starting from a routing at vertex v. Which proves our last implication. □

This means that a particle or an antiparticle can be moved from any vertex of V_0 to any other vertex of V. In particular, any particle can be routed up to a sink if the graph is stopping.

2.4.2 . Cycle pushing and Rotor Walks

Recall that, if there exists a positive (resp., negative) rotor walk r such that $(\rho, \sigma) \overset{r}{\sim} (\rho', \sigma')$ then for all vertex v routed in r, $\rho'^{-}(v)$ (resp., $\rho'(v)$) is the last outgoing arc a particle (resp., an antiparticle) has been routed through to leave v.

Inspired by [41], we define the loop-erased path of a rotor walk at v_0.

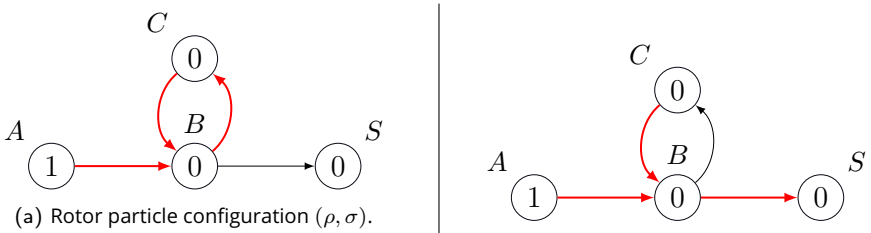
Definition 2.4.8 (Loop-erased path of a rotor walk)**.** *The* loop-erased path *of a maximal rotor walk r at vertex v_0 on a stopping graph is defined as follows. Let p be the path of r. The loop-erased path is obtained by iteratively removing any subsequence of vertices $v_i, v_{i+1}, \ldots, v_j$ in p where $v_i = v_j$, and replacing them with a single occurrence of v_i.*

Proposition 2.4.9 (Cycle push loop-erased path conservation)**.** *Let ρ_1 and ρ_2 be two rotor configurations on a stopping graph such that $\rho_1 \sim \rho_2$ and let σ be a particle configuration. For any two positive maximal rotor walks r_1 and r_2 at $v_0 \in V$ respectively starting from rotor particle configurations (ρ_1, σ) and (ρ_2, σ), the loop-erased path of r_1 and r_2 are the same.*

Proof. To prove the statement, let us consider the case where ρ_2 can be obtained from ρ_1 by a single positive cycle push on a circuit C of $G(\rho_1)$. Studying the case of positive cycle push is enough since if ρ_2 can be obtained from ρ_1 by a single negative cycle push on a circuit C, then ρ_1 can be obtained from ρ_2 by a single positive cycle push on a circuit C and it suffices to exchange the roles of ρ_1 and ρ_2. Let p_1 and p_2 be the paths of r_1 and r_2, respectively. We want to show that their loop-erased paths are equal. If p_1 does not contain

a vertex of C, then $p_1 = p_2$, which proves our statement. Now, let v_c be the first vertex of C that appears in p_1. Since C is a circuit in $G(\rho_1)$, p_1 contains C, and when computing the loop-erased path of r_1, the subsequence C of p_1 will be replaced by v_c. Note that the path of the rotor walk at v_c that proceeds a cycle push on C from (ρ_1, σ) is exactly C. Hence, proceeding this cycle push simply replaces C in p_1 by v_c. Since p_2 is the path of the maximal rotor walk at v_0 where C has been positively cycle pushed, we have that the loop-erased path of r_1 equals the loop-erased path of r_2. This completes the proof. □

Proposition 2.4.9 is illustrated in Figure 2.20.



(a) Rotor particle configuration (ρ, σ).

(b) Let (ρ', σ) be the rotor particle configuration obtained from (ρ, σ) by applying a positive cycle push on the circuit $\{B, C\}$.

Figure 2.20: The path of the maximal rotor walk r_1 at A from (ρ, σ) is $p_1 = A, B, C, B, S$ and the path of the maximal rotor walk r_2 at A from (ρ', σ) is $p_2 = A, B, S$. Since, the subsequence B, C, B appears in p_1, the loop-erased path of r_1 is A, B, S which is also the loop-erased path of r_2.

Now, we prove the unicity of Proposition 2.3.7.

Proof. Note that for an acyclic rotor configuration ρ, the loop erased path of any maximal rotor walk at a vertex v is exactly the directed path with source v in $G(\rho)$. Given a particle configuration σ on a stopping graph. Assume that there are two distinct acyclic rotor configurations ρ_1 and ρ_2 such that $\rho_1 \sim \rho_2$, then, it should exist a vertex $v \in V_0$ such that the loop-erased path of a maximal rotor walk at v from (ρ_1, σ) and (ρ_2, σ) are different which contradicts Proposition 2.4.9. By the fact that any negative cycle push on a set of vertices C in G such that C is a circuit in $G(\rho^-)$ can be simulated by a positive cycle push on C in $G^-(\rho^-)$, we have the same argument on the graph G^-, hence proving that there is an unique rotor configuration ρ in each rotor class such that $G(\rho^-)$ is acyclic. □

2.5 . Legality

This section is dedicated to introducing the concept of *legality* within the context of routing sequences. Legality imposes certain restrictions on the admissible

routing sequences that can be performed starting from a given rotor particle configuration. By imposing these constraints, we aim to explore the properties and behaviors of routing sequences that adhere to specific rules.

Furthermore, this section presents one of the main results derived from [81], which serves as a key motivation for studying the complexity of various problems in Chapter 3. This result establishes a significant connection between the rotor-routing reachability problem and the concept of legality.

2.5.1 . Definitions and fundamental properties

We state first the definitions and properties associated with the legality constraint.

Definition 2.5.1 (Legal Routing Sequence)**.** *A finite or infinite routing sequence r is said to be* legal *if for any two successive rotor particle configurations (ρ_i, σ_i) and $(\rho_{i+1}, \sigma_{i+1})$ of r, there exists a vertex $v_i \in V_0$ such that $\sigma_i(v_i) > 0$ and such that $\mathbf{routing}^+_{v_i}(\rho_i, \sigma_i) = (\rho_{i+1}, \sigma_{i+1})$.*

In order to simplify our notations for the rest of this document, we denote the fact that there exists a legal routing sequence from a rotor particle configuration (ρ, σ) to some (ρ', σ') by $(\rho, \sigma) \to (\rho', \sigma')$. We also denote by $(\rho, \sigma) \xrightarrow{r} (\rho', \sigma')$ the fact that r is a legal routing sequence from (ρ, σ) to (ρ', σ'). So naturally we denote by $(\rho, \sigma) \xrightarrow{R} (\rho', \sigma')$ the fact that R is a legal routing vector from (ρ, σ) to (ρ', σ').

A routing vector is said to be *legal* if there exists a legal routing sequence with this routing vector.

Definition 2.5.2 (Maximal Legal Routing Sequence)**.** *A* maximal legal routing sequence *is a legal routing sequence such that if it is finite, then there is no vertex $v \in V_0$ with $\sigma'(v) > 0$, where σ' is the particle configuration at the end of the routing sequence.*

Lemma 2.5.3 (Rotor Covering)**.** *Given an infinite legal routing sequence, if the graph is strongly connected, all vertices will be routed an infinite number of times.*

Proof. Let r be an infinite legal routing sequence, and let V_r be the set of vertices in V_0 that are routed infinitely many times in r. We want to show that $V_r = V_0$. Suppose for contradiction that $V_r \neq V_0$. Since the graph is strongly connected, there exists a vertex $v \in V_0 \setminus V_r$ that has an incoming arc from a vertex $u \in V_r$. Since u is routed infinitely many times, u sends a particle to v infinitely many times. However, the number of particles on vertices of V_r is finite. Thus, at some point, there will not be any more particle on vertices of V_r and then there are no more routings that are legal on any vertex of V_r which contradicts the assumption that vertices of V_r are routed infinitely many times. Therefore, we must have $V_r = V_0$. □

The introduction of the legality notion implies that it may not be possible to have an infinite legal routing sequence starting from a given rotor particle configuration (ρ, σ). And in that case, we state a few natural questions:

- Does there exist a finite maximal legal routing sequence starting from a rotor-particle configuration (ρ, σ) on a given graph G ? This can be interpreted as asking if all particles reaches a sink vertex or a vertex that has a negative amount of particles on it.
- Given (ρ, σ) and (ρ', σ'), does there exist a legal routing from (ρ, σ) to some (ρ', σ') ?

It is worth noting that the same observation can be made for firing sequences. In fact, it was shown in [16] that from any given starting rotor particle configuration, either every maximal legal firing sequence can be continued indefinitely or every maximal legal firing sequence terminates after finitely many steps. By imposing the legality condition on firing sequences, we restrict the set of possible firing operations to only those that can be performed on a vertex v from a particle configuration σ if and only if $\sigma(v) \geq |\mathcal{A}^+(v)|$. This result naturally extends to legal routing sequence. Thus, to answer our first question, it suffices to determine if there exists a legal routing sequence from (ρ, σ) to some (ρ', σ') with $\sigma'|_{V_0} \leq 0^{|V|}$. It appears that this problem depends on the second question which will be our main focus in this section.

The legal routing sequence is a more general concept than the well-known rotor walk model mostly used in the literature (e.g., [19, 31, 41, 56]) since it allows for negative values in σ on some vertices.

Now, let us turn our attention to the differences in commutativity properties when considering legal routings. It is important to note that the general commutativity property we discussed earlier may not hold when legality constraints are imposed on the routings. This is visually illustrated in Figure 2.21, where the commutativity of routings is altered under the legality constraint.

2.5.2 . Orbits of legal routing

In this section, we investigate the asymptotic behavior of legal routing in a general graph. The main result, Theorem 2.5.11, establishes that if every vertex with a positive number of particles is routed infinitely often, then, after an initial transient phase, the particle configuration can be decomposed into a baseline configuration (a static particle configuration on the vertices such that no vertex can be legally routed) that is independent of the routing order, along with particles that are freely routed on the sink components of the graph (see Figure 2.24 for an illustration). This implies well known results in the literature on the finitness of routing sequences. More specifically, all maximal routing sequences are finite or they are all infinite (Corollary 2.5.6). Furthermore, it also implies the commutativity property of routing sequences which is that in the case where they are finite, the final particle configuration is unique (Corollary 2.5.7).

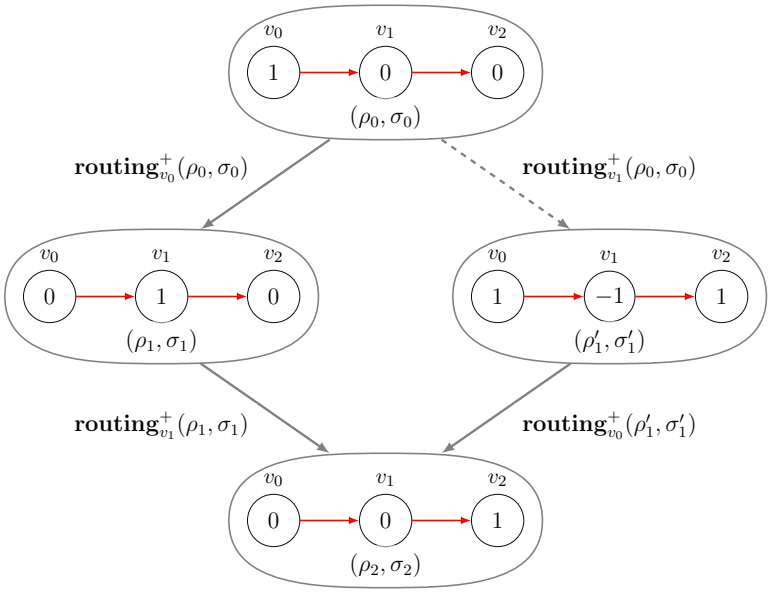


Figure 2.21: Example of two routing sequences with the same routing vector: both v_0 and v_1 are positively routed once. We see that the routing sequence is legal if and only if we proceed the routing at v_0 first.

In what follows, all routing sequences we consider are legal.

Let G be a general rotor graph, and (ρ_0, σ_0) be an arbitrary rotor particle configuration. We denote

$$S(\rho_0, \sigma_0) = \{(\rho, \sigma) : (\rho_0, \sigma_0) \to (\rho, \sigma)\}$$

which contains (ρ_0, σ_0), and

$$S_\infty(\rho, \sigma) = \bigcap_{(\rho', \sigma') \in S(\rho, \sigma)} S(\rho', \sigma')$$

Lemma 2.5.4. *Suppose that $(\rho_0, \sigma_0) \to (\rho, \sigma)$; then $S_\infty(\rho_0, \sigma_0) \subseteq S_\infty(\rho, \sigma)$.*

Proof. If $(\rho, \sigma) \to (\rho', \sigma')$, then by transitivity of the legal routing relation, we have $(\rho_0, \sigma_0) \to (\rho', \sigma')$. This shows that $S(\rho, \sigma) \subseteq S(\rho_0, \sigma_0)$.

Let $(\rho', \sigma') \in S_\infty(\rho_0, \sigma_0)$. By definition of $S_\infty(\rho_0, \sigma_0)$, $(\rho', \sigma') \in S(\rho'', \sigma'')$ for all $(\rho'', \sigma'') \in S(\rho_0, \sigma_0)$ and thus for all $(\rho'', \sigma'') \in S(\rho, \sigma)$. Consequently, $(\rho', \sigma') \in S_\infty(\rho, \sigma)$.

□

Lemma 2.5.5. *Suppose that $(\rho, \sigma) \to (\rho', \sigma')$ and $(\rho, \sigma) \to (\rho'', \sigma'')$; then there exists (ρ^*, σ^*) such that $(\rho', \sigma') \to (\rho^*, \sigma^*)$ and $(\rho'', \sigma'') \to (\rho^*, \sigma^*)$.*

Proof. Consider a legal routing sequence r' from (ρ, σ) to (ρ', σ') and a legal routing sequence r'' from (ρ, σ) to (ρ'', σ''). Let R' and R'' be their associated routing vectors. Define $R^* = \max(R', R'')$, where the maximum is taken

component-wise. Now, consider the rotor-particle configuration (ρ^*, σ^*) reached by such a routing vector from (ρ, σ). We claim that $(\rho', \sigma') \to (\rho^*, \sigma^*)$, and by symmetry, $(\rho'', \sigma'') \to (\rho^*, \sigma^*)$. Refer to Figure 2.22 for an overview of the notations.

Let $\hat{r}$ be the subsequence of r'' obtained by removing the $\min(R'(v), R''(v))$ routings of vertex v, and let $\hat{R}$ be the associated routing vector. By construction $R' + \hat{R} = R^*$. It remains to show that $\hat{r}$ is a legal routing from (ρ', σ') to (ρ^*, σ^*). We denote ϕ as the strictly increasing function that maps indices of elements in $\hat{r}$ to those in r, meaning that the i-th element of $\hat{r}$ is the $\phi(i)$-th element of r.

If $\hat{r}$ is empty, then $(\rho', \sigma') = (\rho^*, \sigma^*)$, and we are done. Otherwise, let v be the first vertex routed in $\hat{r}$. At step $\phi(1)$ of r'', vertex v has been routed the same number of times as in r'. Additionally, the in-neighbor vertices of v have been routed at least as many times in r' as at step $\phi(1)$ of r''. In summary, vertex v has lost $R'(v)$ particles in r' and at step $\phi(1)$ of r'', but it has received more particles from its in-neighborhood in the first case. Since vertex v is legally routable in the latter case, it is also legally routable in the former case.

By replacing (ρ', σ') with the rotor-particle configuration obtained by routing vertex v, we can repeat this reasoning inductively until $\hat{r}$ is empty. This demonstrates that $(\rho', \sigma') \to (\rho^*, \sigma^*)$.

□

This lemma directly establishes the proofs of the following two well-known results.

Corollary 2.5.6. *Given a rotor-particle configuration (ρ, σ), either every maximal legal routing sequence starting from (ρ, σ) is finite or every legal maximal routing sequence is infinite.*

Next result generalizes well-known commutativity results for stopping graph without antiparticule to the legal rotor routing of finite maximal routing w.r.t. the particle configuration that is reached.

Corollary 2.5.7. *Consider a rotor-particle configuration (ρ, σ) where all maximal routings are finite. Assume $(\rho, \sigma) \to (\rho', \sigma')$ and $(\rho, \sigma) \to (\rho'', \sigma'')$, where both routings are maximal. Then $\sigma' = \sigma''$ and $\rho' = \rho''$.*

It is important to note that the assumptions of the corollary are satisfied by a stopping graph regardless of the initial rotor-particle configuration.

Lemma 2.5.8. *Suppose that $(\rho_0, \sigma_0) \to (\rho, \sigma)$; then $S_\infty(\rho_0, \sigma_0) = S_\infty(\rho, \sigma)$.*

Proof. We have already established the inclusion $S_\infty(\rho_0, \sigma_0) \subseteq S_\infty(\rho, \sigma)$ in Lemma 2.5.4. Now, our task is to prove the converse inclusion.

This proof is summarized in Figure 2.23.

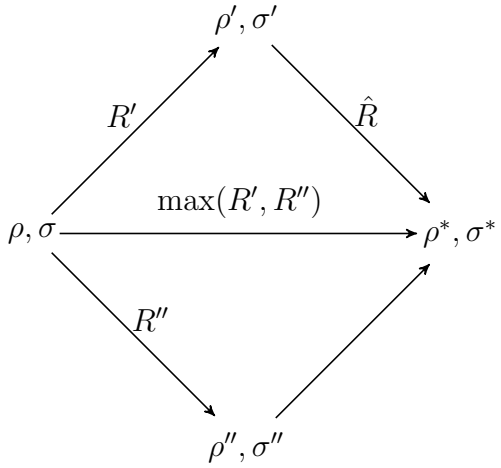


Figure 2.22: Notations used in the proof of Lemma 2.5.5. The depicted rotor-particle configurations represent those involved in the proof. The arcs indicate the existence of a legal routing, with the corresponding routing vector written above each arc.

Let $(\rho', \sigma') \in S_\infty(\rho, \sigma)$. To demonstrate that $(\rho', \sigma') \in S_\infty(\rho_0, \sigma_0)$, we need to show that $(\rho', \sigma') \in S(\rho'', \sigma'')$ for every $(\rho'', \sigma'') \in S(\rho_0, \sigma_0)$.

Let $(\rho'', \sigma'') \in S(\rho_0, \sigma_0)$. Since $(\rho_0, \sigma_0) \to (\rho, \sigma)$ and $(\rho_0, \sigma_0) \to (\rho'', \sigma'')$, we can apply Lemma 2.5.5, which guarantees the existence of (ρ''', σ''') satisfying the following conditions:

- $(\rho, \sigma) \to (\rho''', \sigma''')$
- $(\rho'', \sigma'') \to (\rho''', \sigma''')$

The first condition implies that $(\rho''', \sigma''') \in S(\rho, \sigma)$. Moreover, since $(\rho', \sigma') \in S_\infty(\rho, \sigma)$, we have $(\rho''', \sigma''') \to (\rho', \sigma')$. Combining this with the second condition, we conclude that $(\rho_0, \sigma_0) \to (\rho', \sigma')$ by considering the sequence of routings $(\rho_0, \sigma_0) \to (\rho'', \sigma'')$, $(\rho'', \sigma'') \to (\rho''', \sigma''')$, $(\rho''', \sigma''') \to (\rho', \sigma')$.

□

Lemma 2.5.9. *For any rotor particle configuration (ρ, σ), the set $S_\infty(\rho, \sigma)$ is not empty.*

Proof. Let (ρ, σ) be a rotor-particle configuration. Suppose that $S(\rho_0, \sigma_0) = \{(\rho_i, \sigma_i), i \in \{0, \dots, n\}\}$. For every $i \in \{0, \dots, n\}$, we define the following sets:

$$S^{\leq i}(\rho_0, \sigma_0) = \{(\rho_j, \sigma_j) \mid \forall j \in \{0, \dots, i\}\}$$

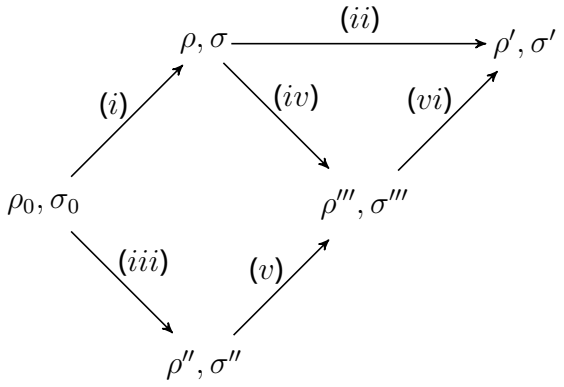


Figure 2.23: Scheme of the proof of Lemma 2.5.8. The depicted rotor-particle configurations represent those involved in the proof. The arcs indicate the existence of a legal routing. This demonstrates the existence of a legal routing from (ρ'', σ'') to (ρ', σ') for any $(\rho'', \sigma'') \in S(\rho_0, \sigma_0)$. In details, (i): assumption of Lemma, (ii): $(\rho', \sigma') \in S_\infty(\rho, \sigma)$, (iii): $(\rho'', \sigma'') \in S(\rho_0, \sigma_0)$, (iv) and (v): use of Lemma 2.5.5, (vi): $(\rho', \sigma') \in S_\infty(\rho, \sigma)$ and $(\rho''', \sigma''') \in S(\rho, \sigma)$.

and

$$S_\infty^{\leq i}(\rho_0, \sigma_0) = \bigcap_{(\rho', \sigma') \in S^{\leq i}(\rho_0, \sigma_0)} S(\rho', \sigma'),$$

such that $S_\infty^{\leq 0}(\rho_0, \sigma_0) = S(\rho_0, \sigma_0)$ and $S_\infty^{\leq n}(\rho_0, \sigma_0) = S_\infty(\rho_0, \sigma_0)$. We prove by induction that $S_\infty^{\leq i}(\rho_0, \sigma_0)$ is non-empty for every $i \in \{0, \dots, n\}$.

This is clearly true for $S_\infty^{\leq 0}(\rho_0, \sigma_0)$.

Suppose it holds for $S_\infty^{\leq i}(\rho_0, \sigma_0)$ with $0 < i < n$. Then,

$$S_\infty^{\leq i+1}(\rho_0, \sigma_0) = S_\infty^{\leq i}(\rho_0, \sigma_0) \cap S(\rho_{i+1}, \sigma_{i+1}).$$

By the induction hypothesis, there exists $(\rho, \sigma) \in S_\infty^{\leq i}(\rho_0, \sigma_0)$. According to Lemma 2.5.5, there exist (ρ'', σ'') such that $(\rho, \sigma) \to (\rho'', \sigma'')$ and $(\rho_{i+1}, \sigma_{i+1}) \to (\rho'', \sigma'')$. Hence, $(\rho'', \sigma'') \in S_\infty^{\leq i+1}(\rho_0, \sigma_0)$.

□

Definition 2.5.10. *Let (ρ_0, σ_0) be a rotor particle configuration, we denote by $F(\rho_0, \sigma_0)$ the non-empty set of particle configurations $\sigma_1 \leq \sigma_0$ (component-wise) such that all routing sequences from (ρ_0, σ_1) are finite. Subsequently, we denote by $F_m(\rho_0, \sigma_0)$ the set of maximal elements in $F(\rho_0, \sigma_0)$.*

Remark that $F(\rho_0, \sigma_0)$ contains $0|_V$ which ensures that it is non-empty.

The following theorem states that, after a transient phase, and regardless of the legal routing chosen, there exists a unique baseline particle configuration above which the remaining particles can be indefinitely routed. Furthermore, these

routings necessarily occur on the sink components of the graph. An example illustrating the computation of the baseline particle configuration is presented in Figure 2.24.

Theorem 2.5.11. *There exists a particle configuration σ_S such that for every $\sigma_1 \in F_m(\rho_0, \sigma_0)$, there exists ρ' such that*

$$S_\infty(\rho_0, \sigma_1) = \{(\rho', \sigma_S)\}$$

and σ_S can be obtained as

$$\sigma_S(v) = \inf\{\sigma(v) : \exists \rho, (\rho, \sigma) \in S_\infty(\rho_0, \sigma_0)\}.$$

In that case, the configuration σ_S is denoted by $\sigma_S(\rho_0, \sigma_0)$.

Proof. Let $\sigma_1 \in F_m(\rho_0, \sigma_0)$. By Lemma 2.5.9, $S_\infty(\rho_0, \sigma_1)$ is non-empty.

Now, let's assume there exist two distinct configurations, (ρ, σ) and (ρ', σ'), in $S_\infty(\rho_0, \sigma_1)$. According to the definition of $S_\infty(\rho_0, \sigma_1)$, we have $(\rho, \sigma) \to (\rho', \sigma')$ and $(\rho', \sigma') \to (\rho, \sigma)$. This implies the existence of an infinite routing sequence starting from (ρ_0, σ_1), where we route from (ρ_0, σ_1) to (ρ, σ), then from (ρ, σ) to (ρ', σ'), and back from (ρ', σ') to (ρ, σ), and so on. However, this contradicts the fact that $\sigma_1 \in F(\rho_0, \sigma_0)$. Hence, $S_\infty(\rho_0, \sigma_1)$ contains only a unique configuration (σ^*, ρ^*) and then $\sigma^* = \sigma_S(\rho_0, \sigma_1)$.

We now prove that $\sigma^* = \sigma_S(\rho_0, \sigma')$ for every $\sigma_1 \leq \sigma' \leq \sigma_0$, which, in particular, implies that $\sigma^* = \sigma_S(\rho_0, \sigma_0)$. If $\sigma_1 = \sigma_0$, then the result is true. Otherwise, there exists a vertex v such that $\sigma_1 + 1_v \leq \sigma_0$, where 1_v represents the configuration that has a value of 1 at vertex v and 0 elsewhere. Let's demonstrate that $\sigma^* = \sigma_S(\rho_0, \sigma_v)$, where $\sigma_v = \sigma_1 + 1_v$. The general case can be handled in a similar manner.

As $(\rho_0, \sigma_1) \to (\rho^*, \sigma^*)$, then $(\rho_0, \sigma_1 + 1_v) \to (\rho^*, \sigma^* + 1_v)$. Since $\sigma_1 \in F(\rho_0, \sigma_0)$, configuration σ^* is not legally routable. But σ^* is maximal in $F(\rho_0, \sigma_0)$, hence the only vertex of $\sigma^* + 1_v$ that is legally routable is v. Such routing leads to a configuration $\sigma^* + 1_w$, with $w \neq v$ since the graph has no loop. By applying this reasoning iteratively, we observe that all reachable particle configurations take the form $\sigma^* + 1_z$ for some vertex z. Then, every particle configuration legally reachable from $(\rho^*, \sigma^* + 1_v)$ is greater than σ^*. In particular, this is the case for every particle configuration in $S_\infty(\rho^*, \sigma^* + 1_v)$, hence for every vertex w:

$$\sigma^*(w) \leq \inf\{\sigma'(w) : \exists \rho, (\rho, \sigma') \in S_\infty(\rho^*, \sigma^* + 1_v)\}$$

On the other hand, in $S_\infty(\rho^*, \sigma^* + 1_v)$, there exist at least two distinct particle configurations of the form $\sigma^* + 1_z$ for different vertices z. Hence, for every vertex w:

$$\sigma^*(w) \geq \inf\{\sigma'(w) : \exists \rho, (\rho, \sigma') \in S_\infty(\rho^*, \sigma^* + 1_v)\}$$

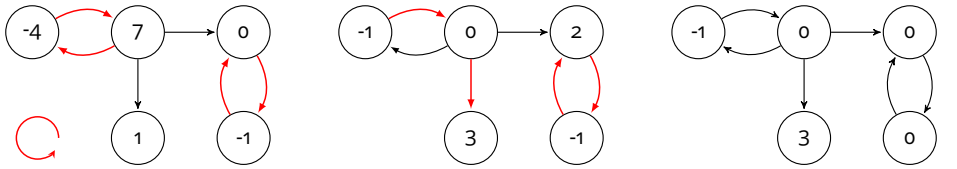
Finally, for every vertex w we have:

$$\begin{aligned}\sigma^*(w) &= \inf\{\sigma'(w) : \exists \rho, (\rho, \sigma') \in S_\infty(\rho^*, \sigma^* + 1_v)\} \\ &= \inf\{\sigma'(w) : \exists \rho, (\rho, \sigma') \in S_\infty(\rho_0, \sigma_v)\} \\ &= \sigma_S(\rho_0, \sigma_v)(w)\end{aligned}$$

where the second equality comes from Lemma 2.5.8, and the last one is the definition of $\sigma_S(\rho_0, \sigma_v)$.

It follows that $S_\infty(\rho_0, \sigma_1)$ does not depend on the choice of σ_1 in $F_m(\rho_0, \sigma_0)$. □

Note that under the conditions of Corollary 2.5.7, which state that all routing sequences are finite, $\sigma_S(\rho, \sigma)$ is the particle configuration reached by any maximal routing sequence starting from (ρ, σ).



(a) Rotor-particle configuration (ρ_0, σ_0).

(b) Rotor-particle configuration obtained after legal routing of the vertex with 7 particles.

(c) Baseline particle configuration $\sigma_s(\rho_0, \sigma_0)$: the sink component with, respectively, 2 and -1 values is replaced by the infimum of the values that can be obtained by every legal routing.

Figure 2.24: Based on Theorem 2.5.11, computation of the baseline particle configuration $\sigma_s(\rho_0, \sigma_0)$.

2.5.3 . Characterization of Rotor Particle configurations reachable by legal routings

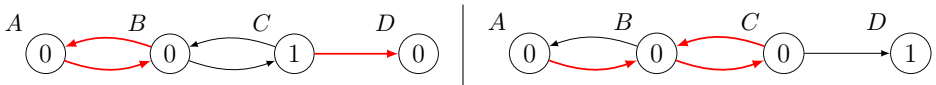
This section presents the main result from [81] which is of significant importance for the rest of this thesis. It was presented in [81] in 2021, and it provides a characterization of the legality of a routing sequence. It will serve as a key tool for providing complexity results in Chapter 3.

Theorem 2.5.12 (Legality Condition Theorem 3.3 from [81]). *Let (ρ, σ) and (ρ', σ') be rotor-particle configurations on G. Then, $(\rho, \sigma) \to (\rho', \sigma')$ if and only if there exists a reduced routing vector R from (ρ, σ) to (ρ', σ') such that we have:*

- $\{v \in V : \sigma'(v) < 0 \text{ and } R(v) > 0\} = \emptyset$

- *For all circuits C in $G(\rho'^{-})$ such that for all vertex $v \in C$, we have $R(v) > 0$, then there exists a vertex $w \in C$ such that $\sigma'(w) > 0$.*

We present one possible usage of this Theorem in Figure 2.25.



(a) Rotor-particle configuration (ρ, σ). (b) Rotor-particle configuration (ρ', σ')

Figure 2.25: Let R be the routing vector $(A : 1, B : 1, C : 1, D : 0)$, since the graph is stopping, the only period vector is $(0, 0, 0, 0)$, thus R is reduced. It is clear that $(\rho, \sigma) \overset{R}{\sim} (\rho', \sigma')$. But, the circuit $\{A, B\}$ belongs to $G(\rho^{-})$ and we have $R(A) > 0$ and $R(B) > 0$ which contradicts the second condition of the previous theorem. Hence, there is no legal routing vector between (ρ, σ) and (ρ', σ').

Corollary 2.5.13 ([81])**.** *The existence of a legal routing sequence r such that $(\rho, \sigma) \xrightarrow{r} (\rho', \sigma')$ can be decided in polynomial time.*

Proof. By Proposition 3.2 of [81], a nonnegative reduced routing vector can be computed in polynomial time, then, it suffices to verify conditions of Theorem 2.5.12. The second condition of Theorem 2.5.12 can be checked by finding all circuits in the graph $G(\rho'^{-})$ and verifying that they satisfy the conditions of the theorem. The number of vertex disjoint circuits in a directed graph is at most $|V|/2$, and for each cycle, we need to check that all vertices in the cycle have positive routing weights and that there is at least one vertex with positive final charge. This can also be done in polynomial time.

Therefore, checking the conditions of Theorem 2.5.12 can be done in polynomial time, given a non-negative routing vector. □

Corollary 2.5.14. *Given a positive rotor particle configuration (ρ, σ) with a particle on each circuit of $G(\rho^{-})$ (there is at least one vertex v of each circuit of $G(\rho^{-})$ such that $\sigma(v) > 0$), if there exists a positive routing vector that transforms some rotor particle configuration (ρ_0, σ_0) into (ρ, σ) then it is a legal routing vector.*

Theorem 2.5.11 also characterizes one particular type of rotor particle configurations, *recurrent configurations* that are positive and have one particle on each circuit as we will see in Corollary 2.5.16.

Definition 2.5.15 (Rotor Particle Recurrent configuration)**.** *On a strongly connected graph G, a rotor particle configuration (ρ, σ) is* recurrent *if there exists a non empty legal routing sequence from (ρ, σ) to (ρ, σ).*

This definition implies not only that G is strongly connected but also that σ is positive by Lemma 2.5.3. Hence, from previous theorem, we can characterize

these configurations on a strongly connected graph. First part of the corollary is in fact Theorem 2.4 of [80] restated in our framework. We give a new proof in our context.

Corollary 2.5.16. *On a strongly connected graph, rotor particle configurations (ρ, σ) with $\sigma(v) \geq 0$ for all $v \in V_0$ and with a particle on each circuit of $G(\rho^-)$ are exactly recurrent configurations. Moreover, if there is a routing vector R from (ρ, σ) to a recurrent rotor particle configuration (ρ', σ'), then R is legal.*

Proof. Let (ρ, σ) be a recurrent rotor particle configuration. By definition, this is equivalent to the fact that $(\rho, \sigma) \to (\rho, \sigma)$. According to Lemma 2.5.3, this sequence must route all vertices in V_0. Therefore, $R(v) > 0$ for every $v \in V_0$. By the second condition of Theorem 2.5.12, this implies that there must be at least one particle on each circuit of $G(\rho^-)$. Now, we show the other implication. Let (ρ, σ) be a positive rotor particle configuration with a particle on each circuit of $G(\rho^-)$. From Corollary 2.5.14, we have that any routing vector that transforms any rotor particle configuration (ρ_0, σ_0) into (ρ, σ) is legal. In particular, there is a legal routing vector from (ρ, σ) to itself which proves that (ρ, σ) is recurrent. □

Now, on stopping graphs, Theorem 2.5.12 implies the two following results that characterize maximal routing sequences for which the ending rotor configuration is acyclic.

Lemma 2.5.17. *Let G be a stopping rotor graph. For any maximal legal routing vector R from (ρ, σ) to some (ρ', σ') with ($\sigma'|_{V_0} = 0|_{V_0}$) such that $R(v) > 0$ for any vertex $v \in V_0$, $G(\rho'^-)$ is acyclic.*

Proof. Suppose that there is a routing vector R from (ρ, σ) to some (ρ', σ') with $\sigma'|_{V_0} = 0|_{V_0}$ such that R is strictly positive. Since the ending configuration is such that $\sigma'|_{V_0} = 0|_{V_0}$, there cannot be a circuit in $G(\rho'^-)$ because it would be a circuit in which all vertices have been routed at least once, which would contradict Theorem 2.5.12. □

Lemma 2.5.18. *Given a rotor configuration ρ on a stopping graph, $G(\rho^-)$ is acyclic if and only if there exists two particle configurations $\sigma, \sigma' \in \mathcal{P}(G)$ with $\sigma(v) > 0$ for all $v \in V_0$ and $\sigma'|_{V_0} = 0|_{V_0}$ such that $(\rho, \sigma) \to (\rho, \sigma')$ with $\sigma'|_{V_0} = 0|_{V_0}$.*

We will prove first that if $G(\rho^-)$ is acyclic, then there exists a particle configuration σ satisfying previous conditions. This relies on some well known results on the action of particle configurations on rotor configurations (see [41]). But, as we focus on the algorithmic aspect of rotor routing in this thesis, we give an alternative proof.

Proof. Let ρ be acyclic, first, consider the strongly connected graph $G' = (V_0' \cup S_0', \mathcal{A}', h, t, \theta)$ such that $V_0' = V_0 \cup S_0$, $S_0' = \emptyset$ and $\mathcal{A}' = \mathcal{A} \cup \mathcal{A}_{S_0}$ with $\mathcal{A}_{S_0}$ being a set with exactly one arc from each sink in S_0 to each vertex of V_0. Let $\rho_{G'}$ be the rotor configuration such that for any vertex $v \in V_0$ we have $\rho_{G'}(v) = \rho(v)$ and for any other vertex $s \in V_0' \setminus V_0$, $\rho_G'(s)$ is any outgoing arc of s. Let p be the primitive period vector of G'.

Consider the firing sequence from $(\rho, 0|_{V_0'})$ to some (ρ, σ') with $\sigma' \in \mathcal{P}_{V'}(G')$ such that each vertex $s \in S_0$ is positively fired exactly $p(s)$ times in r and none of the other vertices are fired. Consider graph G once again. By construction, for any $v \in V_0$, we have $\sigma'(v) > 0$ because p is positive.

Thus, as the graph is stopping, by Lemma 2.5.17, we have that any legal routing sequence from $(\rho, \sigma'|_{S_0})$ to $(\rho', 0|_{V_0})$ in G is such that $G(\rho'^{-})$ is acyclic. We constructed a legal routing sequence from $(\rho, 0|_{V_0})$ to $(\rho', 0|_{V_0})$, thus $\rho \sim \rho'$. By hypothesis we have that $G(\rho^{-})$ is acyclic thus, by Proposition 2.3.7 we have that $\rho' = \rho$.

To prove the other direction, assume that there exists a legal routing sequence r from (ρ, σ) to $(\rho, 0|_{V_0})$ with σ being such that $\sigma(v) > 0$ for all $v \in V_0$. Then, as all vertices have to be routed at least once in order to reach $(\rho, 0|_{V_0})$, by Lemma 2.5.17 we have that $G(\rho^{-})$ is acyclic. □

We also provide an equivalent definition for recurrent rotor configurations on a stopping graph that uses the notion of legality:

Definition 2.5.19 (Recurrent rotor configuration)**.** *A rotor configuration ρ on a stopping rotor graph is* recurrent *if there exists a particle configuration σ such that $|\sigma| > 0$, and there is a legal routing sequence from (ρ, σ) to (ρ, σ') with $\sigma'|_{V_0} = 0|_{V_0}$.*

2.5.4 . Specific results on Stopping Graph

This section examines the finiteness of rotor routing sequences specifically when the graph is stopping. These properties play a crucial role in our investigation of reachability problems in Chapter 3 as well as our analysis of **SP-ARRIVAL** in Chapter 4.

The following lemma is a well-known result regarding legal rotor routing sequences and is also a particular case of Lemma 2.5.5.

Lemma 2.5.20 (Finite number of steps, Lemma 16 in [41])**.** *If G is stopping then any legal routing sequence in G is finite.*

Proof. Suppose that there exists an infinite legal routing sequence on a stopping rotor graph G. Let I be the set of vertices that are routed infinitely many times. Since the graph is stopping, there must exist an arc in G with tail in I and head in $V \setminus I$, which will be visited an infinite number of times. This implies that there is at least one vertex in $V \setminus I$ that is visited an infinite number

of times, which contradicts the assumption that only vertices in I are visited infinitely often. Therefore, there can be no infinite legal routing sequence on a stopping rotor graph. □

3 - ARRIVAL and Reachability Problems

In this document, we call *ARRIVAL* a family of reachability problems with two conditions: the graph is stopping and the initial particle configuration is positive. The question is to determine the particle configuration on sink vertices when all particles of V_0 have been routed up to sinks. Considering results of the previous chapter, this configuration is unique which shows that this problem is well-defined. We distinguish two types of positive configurations, *single particle configuration* where the particle configuration has degree one and *multi particle configuration* where configuration has any positive degree. Using notations of Chapter 2, the first part of this chapter (Section 3.1) presents four problems of *ARRIVAL*. The first one is the version with a single particle on a switching graph studied in [31] that we call from here ***Sw-ARRIVAL*** (referring to the switching graph topology). The second one is a version with a single particle but on a general digraph, that we call ***SP-ARRIVAL*** (where SP stands for "Single Particle"). In the subsequent sections of this document, we introduce a problem referred to as ***MP-ARRIVAL***. This problem is essentially an extension of the **SP-ARRIVAL** problem, allowing for the presence of multiple particles (MP stands for "MultiParticles"). And finally, we introduce the ***Linear ARRIVAL*** problem for which vertices might be routed even if they do not have particles on them. These problems are detailed in Table 3.1. It is known that **Sw-ARRIVAL** belongs to **NP** $\cap$ **co-NP** and we show that this is also the case for our three other problems. Then, we exhibit the connection between **Sw-ARRIVAL** and the rotor-routing reachability problem presented in [81]. We further explain the relationship between these two problems and provide insights into their similarities and differences.

	Legal routing sequence	General routing sequence
Single Particle	**Sw-ARRIVAL** and **SP-ARRIVAL**	**Linear ARRIVAL**
Multiple Particle	**MP-ARRIVAL**	

Table 3.1: Differences between problems of **ARRIVAL**. This family of problems have the following conditions: the graph is stopping and the initial particle configuration is positive. Arrows depict direct reductions between problems (e.g., **SP-ARRIVAL** is a restriction of **MP-ARRIVAL** to instances with a single particle).

In the second and most consequent part of this chapter, we delve into a range of reachability problems within the rotor-routing formalism. These problems are presented in Table 3.2. We explore these problems under different graph topologies, including stopping and strongly connected graphs, and consider the presence or absence of the legality constraint on routings. We compare these various subcases to the **ARRIVAL** problem and provide some complexity results. We begin by presenting our findings on the general reachability problems introduced in [81]. We then focus on a subgroup of problems that are always solvable, meaning that only positive instances exist in the considered subcases. Next, we present our main results on problems related to **ARRIVAL**, including reduction techniques and complexity results. Finally, we discuss subcases for which we have partial results.

Despite most of the material of this chapter is new, findings of this chapter have not yet been published.

3.1 . ARRIVAL

This section is dedicated to establishing the connection between **Sw-ARRIVAL** which is investigated in [31] and the rotor-routing reachability problem introduced in [81]. By establishing this link, we aim to facilitate a comparative analysis of the complexity of the ARRIVAL problem and other general reachability problems in the subsequent section. In [31], the ARRIVAL problem is defined for switching graphs, which are stopping rotor graphs such that each non-sink vertex has exactly two outgoing arcs. That is why we state a more general, yet equivalent version of this problem called **SP-ARRIVAL** on general stopping rotor graphs (SP stands for Single Particle). We will study more deeply this problem in Chapter 4.

3.1.1 . Sw-ARRIVAL and SP-ARRIVAL

In this section we present both problems that are defined with a single particle on the graph. The difference between these two problems is the topology of the underlying graph. Specifically, **Sw-ARRIVAL**$\subset$ **SP-ARRIVAL**. This means that the ARRIVAL problem has the same complexity on switching graphs and general digraphs.

Recall that a stopping rotor graph is a graph G for which there exists a directed path in G from any vertex to at least one sink. Recall that if (ρ, σ) is a rotor particle configuration on G, by Lemma 2.3.12, we have that for any routing sequence from (ρ, σ) to some (ρ', σ'), such that $\sigma'|_{V_0} = 0|_{V_0}$, the particle configuration on the sink vertices $\sigma'|_{S_0}$ is unique.

Definition 3.1.1 (SP-configuration (Positive Single Particle configuration))**.** *Let $u \in V$, a particle configuration σ is a* SP-configuration *on u if for any vertex $v \in V \setminus \{u\}$ we have $\sigma(v) = 0$ and $\sigma(u) = 1$.*

Naturally, if we also consider a rotor configuration, we obtain a *rotor SP-configuration*.

Definition 3.1.2 (Rotor SP-configuration)**.** *A* rotor SP-configuration *is a rotor particle configuration (ρ, σ) such that $\rho \in \mathcal{R}(G)$ and σ is a SP-configuration.*

Remark. *We utilize rotor SP-configurations only in the context of legal routing sequences. Thus, given a rotor SP-configuration on u, all vertices except u have zero particles on them. This means that there is a single legal move possible on any rotor SP-configuration, which is a legal routing at u. Since there is no possible confusion on the position of the unique particle on a Rotor SP-configuration with a particle at vertex u, we use the notation (ρ, u) to refer to such a configuration.*

Note that a legal routing sequence between two rotor SP-configurations is a rotor walk.

Previous remark has motivated the introduction of the following problem in [31]:

Definition 3.1.3 (**Sw-ARRIVAL**)**.** *Let $u \in V$, let (ρ, u) be a rotor SP-configuration on a stopping rotor switching graph G. Given a sink $s \in S_0$,* ***Sw-ARRIVAL*** *is the problem of deciding if there exists a maximal legal rotor walk starting from (ρ, u) such that the particle reaches sink s.*

Obviously a stopping rotor switching graph (see definition in [31]) is also a rotor stopping graph but with a unique rotor order possible. And any rotor stopping graph with a rotor configuration ρ can be transformed in polynomial time into a stopping rotor switching graph with a configuration ρ' by adding $O(|\mathcal{A}^+(v)|)$ vertices for each vertex with outdegree superior to 2 in the idea of Figure 3.1.

This shows that **Sw-ARRIVAL** and the same problem on a general stopping rotor graph (that we call **SP-ARRIVAL**) are equivalent since there is a polynomial reduction from **SP-ARRIVAL** to **Sw-ARRIVAL** and **Sw-ARRIVAL** is a subcase of **SP-ARRIVAL**. Thus we state **SP-ARRIVAL** that is defined on stopping rotor graphs.

Definition 3.1.4 (**SP-ARRIVAL**)**.** *Let $u \in V$, let (ρ, u) be a rotor SP-configuration on a stopping rotor graph G. Given a sink $s \in S_0$,* ***SP-ARRIVAL*** *is the problem of deciding if there exists a maximal legal rotor walk starting from (ρ, u) such that the particle reaches sink s.*

SP-ARRIVAL belongs to NP $\cap$ co-NP for simple graphs as shown in [31], but there is still no polynomial algorithm known to solve it. The proof in [31] relies on the notion of *flow* that we will only introduce in Chapter 4, thus we give an alternative proof with routing vectors to preserve the consistency of this thesis.

First we prove that **SP-ARRIVAL** belongs to **NP**. Let R be a routing vector from (ρ, u) to some rotor SP-configuration (ρ', s) with $s \in S_0$. We show that R is a polynomial certificate for **SP-ARRIVAL**. By Corollary 2.5.7, the rotor

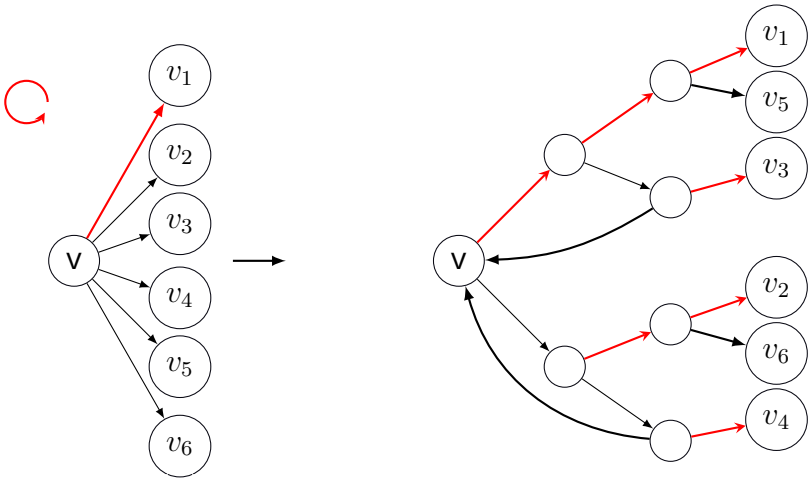


Figure 3.1: Transformation of a rotor graph with a rotor configuration ρ into an equivalent rotor switching graph with a rotor configuration ρ'. One can check that any succession of routing operations on v in the first graph can be simulated by a routing sequence on the second rotor graph. In the general case, any vertex with outdegree k on a general digraph can be turned into a switching subgraph by adding $O(k)$ vertex and $O(k)$ arcs.

configuration ρ' obtained by applying any legal rotor walk with routing vector R is unique, as well as the particle configuration on the sink vertices. Such a rotor configuration can be computed in polynomial time for each $v \in V_0$ by computing the rest of the division $R(v)/|\mathcal{A}^+(v)|$. If the rotor SP-configuration computed from R (by counting the number of routings to the sinks in R) is (ρ', s), it remains to check in polynomial time with Theorem 2.5.12 if $(\rho, u) \to (\rho', s)$. Thus, R is a polynomial certificate for **SP-ARRIVAL**. To prove that **SP-ARRIVAL** belongs to **co-NP**, it suffices to give a routing vector R such that there exists a legal rotor walk from (ρ, u) to some rotor SP-configuration (ρ', s') with $s' \in S_0$ and $s' \neq s$ as this certifies that the ending sink is s'.

The case of simple graphs that are eulerian when we remove the sinks can be solved in time $O(|V+\mathcal{A}|^3)$, since a finite maximal legal rotor walk from (ρ, v) ends in at most $O(|V + \mathcal{A}|^3)$ routings on such a graph (see [85]).

However, no polynomial-time algorithm is currently known to solve **SP-ARRIVAL**. We presented the complexity of this problem in Chapter 1. There we stated that the best upper bound existing for this problem is **UP** $\cap$ **co-UP** [40]. The best known algorithm yet is the subexponential algorithm proposed in [39] which shows that **SP-ARRIVAL** can be decided in time $2^{O(\sqrt{n}\log n)}$ for a graph with n vertices. This result relies on a Tarski fixed point algorithm.

Despite that, even for a path multigraph, a maximal legal rotor walk can be exponential, as shown on Figure 3.2.

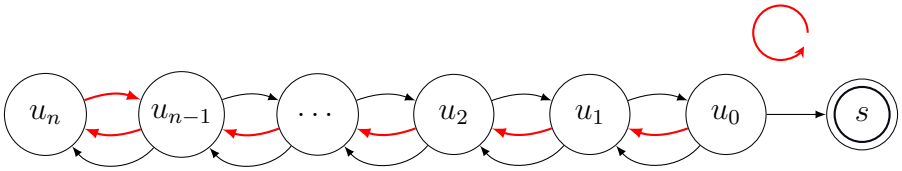


Figure 3.2: Family of path-like multigraphs where a maximal routing sequence can take an exponential number of steps in the number of vertices, here equal to $n+2$. The interior vertices (u_0 to u_{n-1}) have two arcs going left and one going right. The idea behind this example is to construct a binary counter. Routing a particle from u_0 to sink s with the initial configuration ρ drawn with red arcs takes a non-polynomial time considering the anticlockwise rotor ordering on each vertex, depicted by the curved arc in red.

In the example drawn in Figure 3.2, let us consider the number of times the particle will travel from u_i to u_{i+1} during a maximal legal rotor walk from (ρ, u_0). Before the particle reaches the sink s, it needs to visit u_0 exactly three times and so it will travel from u_0 to u_1 exactly two times. Next, for u_1, each time the particle comes from u_0, it will travel two times from u_1 to u_2 before visiting u_0 again. So the number of times the particle travels from u_1 to u_2 is 4. One can check that the number of times a particle starting from u_0 will travel from u_i to u_{i+1} before reaching s is 2^{i+1}.

Nevertheless, we now prove that the number of routings in a legal routing sequence on a stopping rotor graph is bounded.

Lemma 3.1.5. *Let r be a rotor walk from a rotor SP-configuration (ρ, u) on a stopping graph, the particle can travel at most $e^{|\mathcal{A}|/e}$ times through each arc $a \in \mathcal{A}$.*

Proof. According to Lemma 1 in [39], the number of routings on a vertex v in a maximal rotor walk from some (ρ, u) is $\prod_{i=1}^{k} |\mathcal{A}^+(u_i)|$ with k being the length of the shortest path from v to a vertex of S_0. This formula has maximal value if all vertices on the path have $|\mathcal{A}|/n$ outgoing arcs with n the number of vertices of the graph and this is maximal for $n = |\mathcal{A}|/e$ in which case the bound becomes $e^{|\mathcal{A}|/e}$. □

In practice, we will use the following Corollary:

Corollary 3.1.6. *Let r be a legal routing sequence from a rotor particle configuration (ρ, σ) on a stopping graph, each vertex can be routed at most $|\sigma| \cdot \mathcal{MAX}_{v \in V}(\sigma(v)) \cdot e^{|\mathcal{A}|/e}$ in r.*

3.1.2 . MP-ARRIVAL and Linear ARRIVAL

In this subsection, we state the generalization of the **SP-ARRIVAL** problem and its linear version. These differ by the fact that there can be multiple particles on the graph at the same time.

Definition 3.1.7 (**MP-ARRIVAL**). *Given a stopping rotor graph G, a rotor configuration ρ, a positive particle configuration σ, and a particle configuration σ' such that $\sigma'|_{V_0} = 0|_{V_0}$, is there a legal routing sequence from (ρ, σ) to some (ρ', σ')?*

This problem can be interpreted as checking if a maximal legal routing sequence from (ρ, σ) leads to the particle configuration $\sigma'|_{S_0}$ on sink vertices.

MP-ARRIVAL belongs to **NP** $\cap$ **co-NP** since we can check in polynomial time if a routing vector is legal.

The problem that is solved in [81] is stated as the reachability problem from a rotor particle configuration (ρ, σ) to a rotor particle configuration (ρ', σ') by a legal routing sequence.

Clearly, **MP-ARRIVAL** can also be seen as a reachability problem between two positive rotor particle configurations but where ρ' is not given.

Thus, it seems like an interesting way of studying it by comparing its complexity to the one of other rotor-routing reachability problems. The main purpose of this chapter is to give some perspective on the different reachability problems that can be studied on rotor graphs and how **MP-ARRIVAL** inserts in Table 3.2.

Linear ARRIVAL

We define here a new linear version of **MP-ARRIVAL**, without the legality constraint. Contrary to **MP-ARRIVAL**, in **Linear ARRIVAL**, vertices might be routed even if they happen to be negative at some point in the routing sequence.

Definition 3.1.8 (**Linear ARRIVAL**). *Given a stopping rotor graph G, a rotor configuration ρ, a particle configuration σ which is positive on V_0, and a particle configuration σ' such that $\sigma'|_{V_0} = 0|_{V_0}$, is there a routing sequence from (ρ, σ) to some (ρ', σ')?*

Observe that, contrary to a positive instance of **MP-ARRIVAL**, in a positive instance of **Linear ARRIVAL**, there might be several different ending rotor configurations (see Corollary 2.5.7 and Lemma 2.3.12). Nevertheless, we show that **MP-ARRIVAL** and **Linear ARRIVAL** are equivalent, in the sense that given ρ, σ and σ' there is a solution to **Linear ARRIVAL** if and only if there is a solution to **MP-ARRIVAL**.

Lemma 3.1.9. *Given a stopping rotor graph G, a rotor configuration ρ, a positive particle configuration σ, and a particle configuration σ' such that $\sigma'|_{V_0} = 0|_{V_0}$,*

$$\exists \rho' \in \mathcal{R}(G), (\rho, \sigma) \sim (\rho', \sigma') \Leftrightarrow \exists \rho'' \in \mathcal{R}(G), (\rho, \sigma) \to (\rho'', \sigma')$$

Proof. It is direct that if $(\rho, \sigma) \to (\rho'', \sigma')$, then $(\rho, \sigma) \sim (\rho'', \sigma')$. Conversely, assume that $(\rho, \sigma) \sim (\rho', \sigma')$. Since, $\sigma|_{V_0}$ is positive, there exists a legal routing sequence to a particle configuration σ'_t such that $\sigma'_t|_{V_0} = 0|_{V_0}$. Hence, by Lemma 2.3.12, we have that $\sigma' = \sigma'_t$ which proves that there exists a rotor configuration ρ'' such that $(\rho, \sigma) \to (\rho'', \sigma')$. □

From this lemma, we have the following theorem.

Theorem 3.1.10 (**Linear ARRIVAL** Equivalence)**.** *Given a stopping rotor graph, a rotor configuration ρ, a positive particle configuration σ, and a particle configuration σ', the decision problems **MP-ARRIVAL** and **Linear ARRIVAL** have the same truth value.*

3.2 . Reachability Problems Chart

In the previous chapter, we discussed various properties related to the legality and presence of sinks in rotor-routing graphs. This section presents a more detailed exploration of reachability problems in rotor-routing, considering two different frameworks: a linear case and a case where only legal routing sequences are considered. Our results in both frameworks depend mainly on the presence of accessible sinks in the graph, and hence, we divide both frameworks into two subcases: one where $S_0 = \emptyset$ and the graph is strongly connected, and another where $S_0 \neq \emptyset$ and the graph is stopping. Recall that S_0 is the set of vertices with no outgoing arcs.

We summarize our complexity results in Table 3.2 where:

- The first column describes the problem we are interesting in. Each problem is denoted with $\overset{\sim}{\rightarrow}$ since we study both the linear and the legal case.

- The second column called *Linear Case* describes the framework where we consider general routing sequences (no legality). This column is split into two semicolons, the first one where we consider strongly connected graphs, and the second one where we consider stopping graphs (with $S_0 \neq \emptyset$).

- The third column describes the framework where we only consider legal routing sequences. This column is also split into two semicolons, one for strongly connected graphs and one for stopping graphs.

Each problem presented in Table 3.2 is a decision problem that poses the question of whether there exists a routing sequence for given rotor configurations ρ and ρ', as well as particle configurations σ and σ'. We sometimes add some extra conditions on the routing sequence. Possibly, some problems might have « $*$ » in their initial parameters, for example $(\rho, *) \overset{\sim}{\rightarrow} (\rho', *)$. This means that we ask if there exist two particles configurations σ and σ' such that there is a routing sequence from (ρ, σ) to (ρ', σ'). For some problems, the complexity results have been previously stated in the literature, and the paper where they were first stated is listed next to them.

For example, with this formalism, given a rotor configuration ρ, **SP-ARRIVAL** can be written as $(\rho, u) \rightarrow (*, s)$ with $u \in V$ and $s \in S_0$.

Legality Constraint	**Linear Case**		**Legal Case**	
Graph Topology	Strongly Connected	Stopping	Strongly Connected	Stopping
Firing Sequence Only: $(\rho,\sigma) \overset{\sim}{\to} (\rho,\sigma')$	**P** see [52]		**co-NP**† see [52]	
$(\rho,\sigma) \overset{\sim}{\to} (\rho',\sigma')$	**P** see [80]		**P** see [81]	
$(\rho,\sigma) \overset{\sim}{\to} (*,\sigma')$	**True** iff same degree	Equivalent to **MP-ARRIVAL**	Equivalent to **MP-ARRIVAL**†	
$(*,\sigma) \overset{\sim}{\to} (\rho',\sigma')$	**True** iff same degree	Equivalent to **MP-ARRIVAL**	**NP**†	
$(*,\sigma) \overset{\sim}{\to} (*,\sigma')$	**True** iff same degree	**NP-Complete**	**NP-Complete**†	
$(\rho,\sigma) \overset{\sim}{\to} (\rho',*)$	**True**		**NP**†	
$(\rho,*) \overset{\sim}{\to} (\rho',\sigma')$	**True**		**NP**†	
$(\rho,*) \overset{\sim}{\to} (\rho',*)$	**True**			

Table 3.2: Our complexity results of problems described in this chapter. The word *True* indicates the cases where for any instance of the problem, there always exists a routing sequence (i.e., the answer to the reachability problem is *True*). The † indicates the cases where we can give better complexity results with some additional constraints.

Remark. *The notion of equivalence between problems A and B, as presented in Table 3.2 and throughout this chapter, signifies that there is a polynomial reduction from A to B, and there is a polynomial reduction from B to A.*

3.3 . General Reachability Problems

3.3.1 . Problem $(\rho,\sigma) \overset{\sim}{\to} (\rho,\sigma')$ by a firing sequence [52]

We start this section by quickly presenting the results of [52] in our framework since there will be used to prove our results for general routing sequences.

Note that here, we are considering the problem of a firing sequence between two rotor particle configurations with the same rotor configuration. Moreover, our results for this problem do not rely on the presence of sinks or not.

3.3.1.1 $(\rho,\sigma) \sim (\rho,\sigma')$ by a firing sequence

For this first problem, we recall that a firing operation on a vertex v is the action of proceeding successively $|\mathcal{A}^+(v)|$ positive (resp., negative) routings on v. Lemma 2.3.11 states that there is a polynomial algorithm that decides whether there exists a firing vector from (ρ,σ) to (ρ,σ') and if a firing vector exists then the algorithm returns it. Thus, as we have no constraint of legality, we have the following result:

Lemma 3.3.1. *Problem $(\rho,\sigma) \sim (\rho,\sigma')$ is in **P**.*

3.3.1.2 $(\rho, \sigma) \to (\rho, \sigma')$ by a legal firing sequence

Recall that a legal firing sequence is a firing sequence in which each vertex can only be fired if it has more particles on it than its outdegree.

Thus, if we add the legality constraint, computing a non-negative firing vector from (ρ, σ) to (ρ, σ') in polynomial time (if one exists) does not guarantee the existence of a legal firing sequence with that firing vector, as illustrated in Figure 3.3.

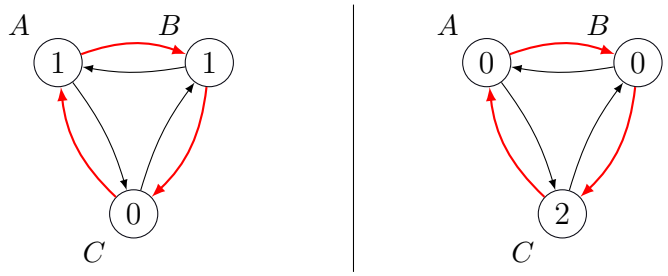


Figure 3.3: Let ρ be the rotor configuration depicted by the red arcs. Let σ be the particle configuration illustrated on the figure on the left. Let σ' be the particle configuration illustrated on the figure on the right. There exists a positive firing vector R such that $R(A) = 1, R(B) = 1, R(C) = 0$ that transforms (ρ, σ) into (ρ, σ') but there is no legal sequence to do it as there are no legal firing doable from (ρ, σ).

Unfortunately, the conditions provided by Theorem 2.5.12 are necessary but not sufficient to certify the legality of the firing sequence. Nevertheless, it has been shown in [52] that this problem belongs to **co-NP**. Finding a polynomial algorithm to solve this problem seems complicated as it is proved in [81] that if there existed a polynomial algorithm to solve this problem, then another problem called the firing halting problem, which has been proven to be **NP-Complete** in [34], would be in **co-NP**.

Nevertheless, note that there exists a strongly polynomial algorithm to solve this problem on Eulerian digraphs as shown in [52].

3.3.2 . Reachability problem of [81] $(\rho, \sigma) \xrightarrow{\sim} (\rho', \sigma')$

Once again, results of this subsection do not depend on the presence of sinks or not.

For simplicity of the following proofs, we define the operation of *aligning* two rotor configurations.

Definition 3.3.2 (Aligning Vectors)**.** *Given two rotor configurations ρ and ρ' on a graph G. For any vertex $v \in V_0$, let x_v^+ be the lowest positive exponent such that $\theta_v^{x_v^+}(\rho(v)) = \rho'(v)$ and let x_v^- be the lowest positive exponent such that $\theta_v^{-x_v^-}(\rho(v)) = \rho'(v)$. Let $R^+(\rho, \rho')$ be the routing vector with values $R^+(\rho, \rho')(v) =$*

x_v^+ *for* $v \in V_0$ *and let* $R^-(\rho, \rho')$ *be the routing vector with values* $R^-(\rho, \rho')(v) = x_v^-$ *for* $v \in V_0$. *We call these routing vectors positive (resp., negative)* aligning vectors *from* ρ *to* ρ'.

Note that processing a routing sequence with routing vector $R^+(\rho, \rho')$ from a rotor particle configuration (ρ, σ) leads to some rotor particle configuration (ρ', σ').

3.3.2.1 Problem $(\rho, \sigma) \sim (\rho', \sigma')$

First, we compute an aligning vector from ρ to ρ' in polynomial time by performing at most $|\mathcal{A}^+(v)|$ operations for each vertex $v \in V_0$, which adds up to a total of $|\mathcal{A}|$ operations. Next, we obtain the problem $(\rho', \sigma) \sim (\rho', \sigma')$ which is the firing sequence problem presented above. Therefore we have the following lemma:

Lemma 3.3.3. *Problem* $(\rho, \sigma) \sim (\rho', \sigma')$ *is in* ***P****.*

3.3.2.2 Problem $(\rho, \sigma) \to (\rho', \sigma')$

To determine if there is a legal routing sequence from (ρ, σ) to (ρ', σ'), we follow the same procedure as in the linear case. Firstly, we solve problem $(\rho, \sigma) \sim (\rho', \sigma')$ in the general case which can be done in polynomial time. If the problem has a solution, we apply Theorem 2.5.12 to verify whether this routing sequence is legal, which can also be done in polynomial time. Therefore, this proves the following lemma:

Lemma 3.3.4. *Problem* $(\rho, \sigma) \to (\rho', \sigma')$ *is in* ***P****.*

There is a subcase of this problem that is worth mentioning: we want to determine if there exists a cycle push sequence from (ρ, σ) to (ρ', σ).

3.3.2.3 Problem $(\rho, \sigma) \overset{\sim}{\to} (\rho', \sigma)$

Cycle push sequence $(\rho, \sigma) \sim (\rho', \sigma)$

If $\sigma \neq \sigma'$, there is no such sequence as it does not change the particle configuration. If $\sigma = \sigma'$ and if $(\rho, \sigma) \sim (\rho', \sigma)$, by Theorem 2.3.5, there also exists a cycle push sequence from (ρ, σ) to (ρ', σ). Therefore, to determine the existence of a cycle push sequence from (ρ, σ) to (ρ', σ'), it suffices to determine if there is a routing sequence from (ρ, σ) to (ρ', σ), which can be done in polynomial time by Lemma 3.3.3.

Cycle push sequence $(\rho, \sigma) \to (\rho', \sigma)$

The first step is to decide whether there exists a legal routing sequence from (ρ, σ) to (ρ', σ), which can be done in polynomial time by Lemma 3.3.4. If there exists a legal routing sequence, it is also positive, and we can apply Theorem 2.3.5 to decompose it into cycle pushes. This gives us the following proposition.

Proposition 3.3.5. *The problem of finding a legal cycle push sequence from* (ρ, σ) *to* (ρ', σ') *belongs to* ***P****.*

3.4 . Properties for problems with a missing input on strongly connected graphs in the linear case

This section presents simple results on problems with at least one missing input on strongly connected graphs. First, note that all problems with a missing input clearly belongs to **NP**. As an example, suppose that problem $(*, \sigma) \to (\rho', \sigma')$ has a solution, it suffices to give a rotor configuration ρ and one can check in polynomial time if there exists a legal routing sequence from (ρ, σ) to (ρ', σ'). Thus, rotor configuration ρ is a polynomial certificate for this problem. Indeed, the missing input is a polynomial certificate for any problem as we can check if $(\rho, \sigma) \sim (\rho', \sigma')$ in polynomial time by Lemma 3.3.3 and if $(\rho, \sigma) \to (\rho', \sigma')$ in polynomial time by Lemma 3.3.4 depending on the context.

3.4.1 . Problems where we can choose the rotor configuration(s)

In this section, we present results that are applicable to three different problems: $(\rho, \sigma) \sim (*, \sigma')$, $(*, \sigma) \sim (\rho', \sigma')$, and $(*, \sigma) \sim (*, \sigma')$.

First we show the following lemma:

Lemma 3.4.1. *Problems* $(\rho, \sigma) \sim (*, \sigma')$ *and* $(*, \sigma) \sim (\rho', \sigma')$ *are equivalent.*

Proof. It suffices to change the sign of any routing sequence from $(\rho, \sigma) \sim (*, \sigma')$ to obtain a routing sequence from $(*, \sigma') \sim (\rho, \sigma)$ □

If $|\sigma| \neq |\sigma'|$, then since the routing operation does not change $|\sigma|$, there does not exist a routing sequence for any of the three problems. Next, suppose that $|\sigma| = |\sigma'|$.

For the three problems, let $\Sigma^- = \{v \in V \mid \sigma'(v) - \sigma(v) < 0\}$ be the set of vertices that should give particles (or receive antiparticles) if there exists a routing sequence, and let $\Sigma^+ = \{v \in V \mid \sigma'(v) - \sigma(v) > 0\}$ be the set of vertices that should receive particles (or give antiparticles) in the routing sequence. Take any $v^- \in \Sigma^-$ and $v^+ \in \Sigma^+$. Since there is no restriction on the routing sequence and since the graph is strongly connected, by Lemma 2.4.7, there exists a positive routing sequence that transfers a particle from v^- to v^+ independently of the initial rotor configuration.

By repeating this process enough time until there are no more vertices in Σ^+ and Σ^-, we can construct a routing sequence $(\rho, \sigma) \sim (*, \sigma')$. Obviously, this also shows that there always exists an initial rotor configuration ρ such that there is a routing sequence $(*, \sigma) \to (*, \sigma')$. And, since $(\rho, \sigma) \sim (*, \sigma')$ and $(*, \sigma) \sim (\rho', \sigma')$ are equivalent for the general case, there always exists a routing sequence for these three problems.

3.4.2 . Problems where we can choose the particle configuration(s)

In this subsection, we present results that are applicable to three different problems: $(\rho, *) \sim (\rho', \sigma')$, $(\rho, \sigma) \sim (\rho', *)$ and $(\rho, *) \sim (\rho', *)$. Note that problems $(\rho, *) \sim (\rho', \sigma')$ and $(\rho, \sigma) \sim (\rho', *)$ are equivalent in the linear case. These problems share the common feature that at least one of the particle configurations σ or σ' is not fixed beforehand.

For problems $(\rho, \sigma) \sim (\rho', *)$, and $(\rho, *) \sim (\rho', *)$ it suffices to align ρ and ρ' using any sequence with a routing vector that is an aligning vector (positive or negative). Since problems $(\rho, *) \sim (\rho', \sigma')$ and $(\rho, \sigma) \sim (\rho', *)$ are equivalent, there always exists a routing sequence for these three problems.

Note that the presence of sinks does not affect these results, hence we have the following proposition.

Proposition 3.4.2. *Problems $(\rho, \sigma) \sim (\rho', *)$, $(\rho, *) \sim (\rho', *)$ and $(\rho, *) \sim (\rho', \sigma')$ always have a solution in the linear case both for strongly connected and stopping graphs.*

3.5 . Problem $(*, \sigma) \xrightarrow{\sim} (*, \sigma')$

In this section, we will discuss the results for problem $(*, \sigma) \sim (*, \sigma')$ and we will show various equivalences and complexity results summarized in Figure 3.4.

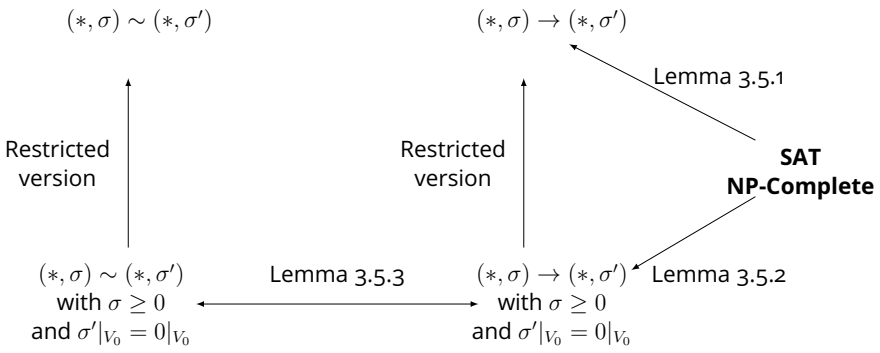


Figure 3.4: Equivalence and Complexity Results for problem $(*, \sigma) \xrightarrow{\sim} (*, \sigma')$. An arrow represents a polynomial reduction from the problem at the tail to the problem at the head.

A pretty similar problem has already been studied in [77]. It is presented as a one player version of **SP-ARRIVAL** (which we will study more deeply in Chapter 4) where the initial rotor configuration is partially fixed. It could be restated with our formalism by $(\rho(U, *), \sigma) \to (*, \sigma')$ with $\rho(U, *)$ being a partial rotor configuration on $U \subset V$, with σ being a SP-configuration and σ' being such

that $\sigma'|_{V_0} = 0|_{V_0}$. The corresponding reachability question is: does there exist a partial rotor configuration $\rho|_{V_0 \setminus U}$ and a rotor configuration ρ' such that $(\rho(U, *) \cup \rho|_{V_0 \setminus U}, \sigma) \to (\rho', \sigma')$. Despite being a problem with a single particle on a stopping graph, this problem has been shown to be **NP-Complete**. But, this problem is different from $(*, \sigma) \xrightarrow{\sim} (*, \sigma')$ since we consider particle configurations with several particles. While we have no direct reduction between these two problems, we will nevertheless show in this section that $(*, \sigma) \xrightarrow{\sim} (*, \sigma')$ is **NP-Complete** in the stopping case.

3.5.1 . Gadget

For the following reduction of this chapter we need a polynomial length gadget that simulates a vertex v with k arcs (v, a) and 2^N arcs (v, b) with N being a fixed integer parameter. The first k particles entering the gadget will be routed towards vertex a and the 2^N particles entering after will be routed towards vertex b. (Figure 3.5)

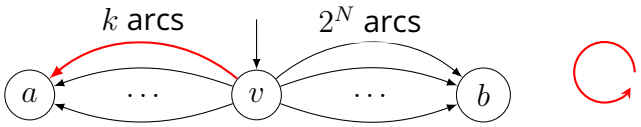


Figure 3.5: First k particles entering v will be routed towards a and 2^N following particles will be routed towards b.

To achieve that, we replace v with a slightly modified version of the graph presented in Figure 3.2 as in Figure 3.6.

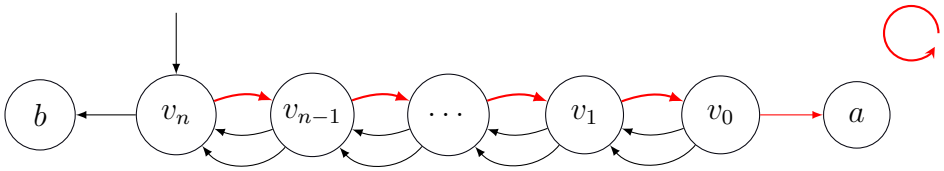


Figure 3.6: Polynomial length gadget that routes the first particle arriving at v_n towards a and the 2^n following particles are routed towards b. We denote such a gadget by $G_{a,b}(1, 2^n)$.

Our previous example shows a gadget that routes only the first particle towards a vertex a and then the following particles to a vertex b. In order to obtain a gadget that simulates a vertex v with k arcs (v, a) and 2^N arcs (v, b) we proceed as in Figure 3.7.

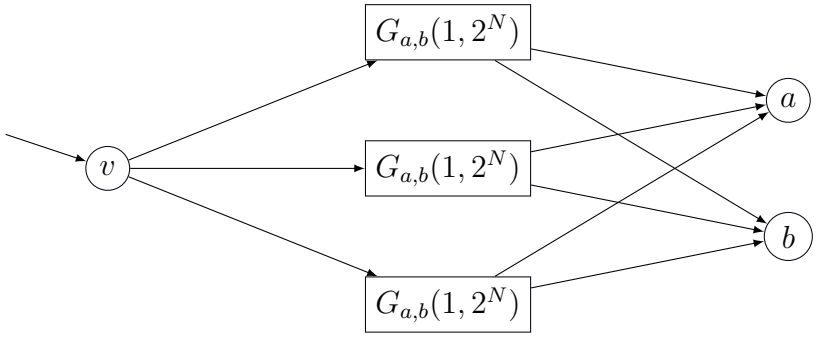


Figure 3.7: Polynomial length gadget that routes the k first particles arriving at v towards a and the following particles towards b with k being the number of gadgets $G_{a,b}(1, 2^N)$. We denote such a gadget by $G_{a,b}(k, k \cdot 2^N)$.

In this section, we will use this gadget to force the k first particles entering a vertex v to be routed to a specific neighbour of v and then the 2^N following particles routed to follow the initial rotor order at v as illustrated in Figure 3.8.

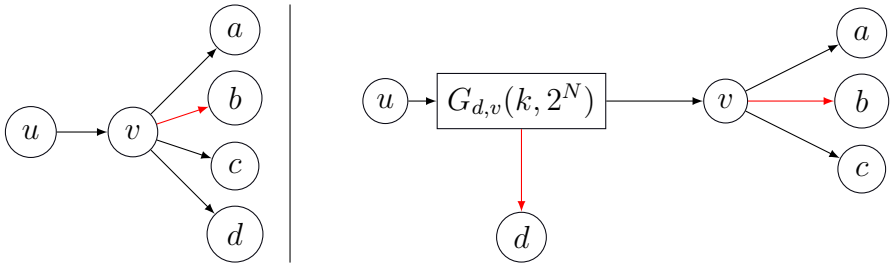


Figure 3.8: Construction that forces the k first particles initially routed at v to reach d and the 2^N next particles to follow the original relative rotor order at v without d.

3.5.2 . Proof that $(*, \sigma) \to (*, \sigma')$ and $(*, \sigma) \sim (*, \sigma')$ are NP-Complete

This subsection proves that both $(*, \sigma) \to (*, \sigma')$ and $(*, \sigma) \sim (*, \sigma')$ are **NP-Complete**. First we prove this result in the stopping case and then in the strongly connected case. Consider the problem **SAT** whose complexity was shown to be **NP-Complete** by Cook's Theorem in [22].

3.5.2.1 Case where the graph is stopping

We recall that the CNF version of the SAT problem is a satifiability problem of a boolean formula of the form: $F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ with each clause C_j being like $C_j = (l_{j_1} \vee l_{j_2} \vee \cdots \vee l_{j_k})$ with literals $l_{j_p} \in \{x_i, \bar{x_i}, i \leq n\}$ and with each x_i being a boolean variable. For such an instance the question is whether there exists an assignment of $(x_1, x_2, \ldots, x_n)$ such that F is satisfied. Let $\Gamma^+(x_i)$ (resp., $\Gamma^-(x_i)$) be the application that associates to each variable the number of times variable x_i appears positively (resp., negatively) in the boolean formula F. And let $\Gamma(x_i) = \textbf{max}(\Gamma^+(x_i), \Gamma^-(x_i))$.

This subsection presents the proof of the following Lemma:

Lemma 3.5.1. *Problem $(*, \sigma) \to (*, \sigma')$ is **NP-Complete** on stopping graphs.*

We propose the following reduction to show that any instance of the CNF **SAT** problem can be solved by solving an instance of $(*, \sigma) \to (*, \sigma')$. We detail the reduction for problem $(*, \sigma) \to (*, \sigma')$ when the graph has sinks but the case where the graph is sinkless and strongly connected is very similar as we will explain later.

We construct a directed rotor graph G_F as follows. Each boolean variable x_i of the instance of the **SAT** problem we are considering is represented in G_F by a vertex X_i. Consider a global sink S. Each clause C_j is represented in G_F by a vertex C_j. For each clause C_j, for each variable $x_i \in C_j$ we add an arc from X_i to C_j. Let us add one arc from each clause C_j to S. Then, let us add $\Gamma(x_i)$ arcs from X_i to S. Now, for each variable x_i, let us add two sink vertices $\textbf{true}_i$ and $\textbf{false}_i$. Add one arc from X_i to $\textbf{true}_i$ and one arc from X_i to $\textbf{false}_i$. The rotor

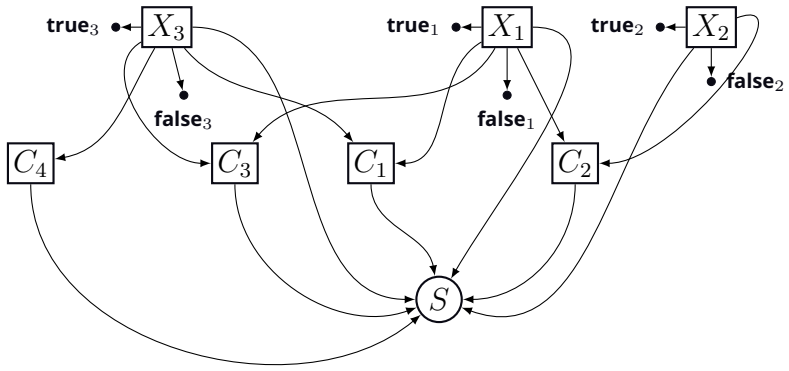


Figure 3.9: Construction of the graph G_F for the boolean formula $F = (x_1 \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2}) \wedge (x_1 \vee x_3) \wedge x_3$. Note that this construction is for a 2-SAT problem for better understanding, but the idea is the same for the general **SAT** problem.

order at each C_i only has one arc so there is no choice. A rotor order at X_i is a rotor order compatible with the following partial order:

1. All arcs from X_i to a clause C_j where x_i appears positively
2. $\mathbf{max}(\Gamma^-(x_i) - \Gamma^+(x_i), 0)$ arcs from X_i towards S
3. The arc from X_i to $\mathbf{false}_i$
4. All arcs from X_i to a clause C_j where x_i appears negatively
5. $\mathbf{max}(\Gamma^+(x_i) - \Gamma^-(x_i), 0)$ arcs from X_i towards S
6. The arc from X_i to $\mathbf{true}_i$

Note that there are exactly $\Gamma(x_i)$ arcs while summing (1) and (2) as well as when summing (4) and (5).

We give an example of this construction in Example 3.9.
Let σ be the particle configuration of G_F such that for all $i \leq n$ and for all $j \leq m$, $\sigma(C_j) = -1$, $\sigma(X_i) = \Gamma(x_i) + 2$ and $\sigma(\mathbf{true}_i) = \sigma(\mathbf{false}_i) = \sigma(S) = 0$.
Figure 3.10 shows an example of σ on the graph G_F of the previous example.

Let σ' be the particle configuration on G_F for which $\sigma'(X_i) = \sigma'(C_j) = 0$, $\sigma'(\mathbf{true}_i) = \sigma'(\mathbf{false}_i) = 1$ for all $i \leq n$ and $j \leq m$ and $\sigma'(S) = \Sigma_{i \leq n}\Gamma(x_i) - m$. This is a particle configuration where all particles are on the sink vertices.

We show first that if there exist ρ and ρ' two rotor configurations on G_F such that there is a legal routing sequence r from (ρ, σ) to (ρ', σ') then there exists an assignment of $(x_1, x_2, \ldots, x_n)$ such that F is satisfied.

Note that as there are only $\Gamma(x_i) + 2$ particles on X_i, and as there are $\Gamma(x_i)$ arcs in the rotor order at X_i between the arc directed towards $\mathbf{false}_i$ and the arc

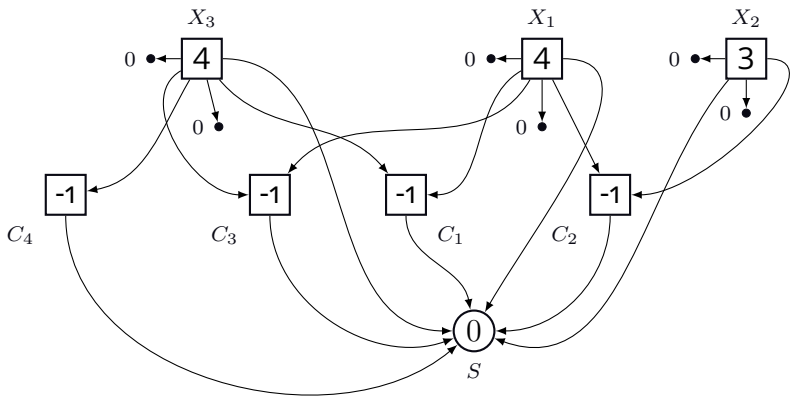


Figure 3.10: Particle configuration σ on the graph G_F for the boolean formula $F = (x_1 \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2}) \wedge (x_1 \vee x_3) \wedge x_3$.

directed towards **true**$_i$, since the graph is acyclic, in order to reach configuration σ' such that $\sigma'(\textbf{true}_i) = \sigma'(\textbf{false}_i) = 1$ from a rotor particle configuration (ρ, σ), we should have $\rho(X_i)$ equals to one of these two arcs. Thus, the choice between those arcs represents the valuation of variable x_i, either *true* or *false*. Obviously, if there exists a routing sequence r, then all vertices C_j receive at least one particle in the sequence. Given what we have already said on ρ, this only happens if there exists an assignment of $(x_1, x_2, \ldots, x_n)$ such that each clause is true. Hence, we showed that if there is a solution to problem $(*, \sigma) \to (*, \sigma')$, then F is satisfiable. Figure 3.11 presents a rotor configuration on Figure 3.10 that proves that $(*, \sigma) \to (*, \sigma')$ has a solution and so that the formula F is satisfiable.

Now, it remains to show that if there is a solution to the SAT problem, then there is a solution to problem $(*, \sigma) \to (*, \sigma')$ on the graph G_F with σ and σ' defined as before.

Assume that there is a solution to the SAT problem, then, there exists a valuation ϕ of $(x_1, x_2, \ldots, x_n)$ that satisfies F.

Let ρ be a rotor configuration such that $\rho(X_i) = \textbf{true}_i$ if $\phi(x_i) = True$ and $\rho(X_i) = \textbf{false}_i$ if $\phi(x_i) = False$. By construction, we have $(\rho, \sigma) \to (*, \sigma')$ on G' which proves that $(*, \sigma) \to (*, \sigma')$ has a solution.

This shows that $(*, \sigma) \to (*, \sigma')$ is **NP-hard** and as we stated at the beginning of this chapter, this problem also belongs to **NP**. Thus, we have that $(*, \sigma) \to (*, \sigma')$ is **NP-Complete**.

Note that the graph constructed by this reduction is simple and acyclic, hence even for really simple class of graphs this problem remains **NP-Complete**.

Next, we show that a restricted version of $(*, \sigma) \to (*, \sigma')$ is also **NP-Complete** as it will be an important result for the proof that $(*, \sigma) \sim (*, \sigma')$ is **NP-Complete**.

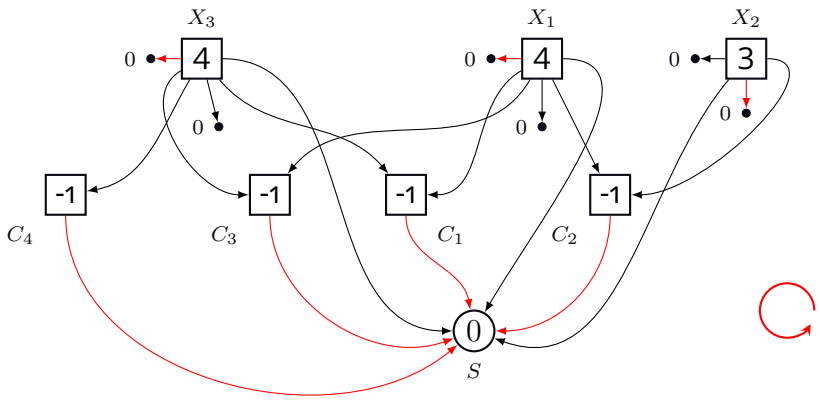


Figure 3.11: The rotor configuration depicted in red solves problem $(*, \sigma) \rightarrow (*, \sigma')$ obtained by our construction from the boolean formula $F = (x_1 \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2}) \wedge (x_1 \vee x_3) \wedge x_3$. This shows that F is satisfiable.

Proof that $(*, \sigma) \rightarrow (*, \sigma')$ is NP-Complete on stopping graphs when $\sigma \geq 0$ and $\sigma'|_{V_0} = 0|_{V_0}$

The purpose of this paragraph is to show that the previous reduction holds for problem $(*, \sigma) \rightarrow (*, \sigma')$ where $\sigma \geq 0$ and $\sigma'|_{V_0} = 0|_{V_0}$. We will further use this result to show that $(*, \sigma) \sim (*, \sigma')$ is **NP-Complete**.

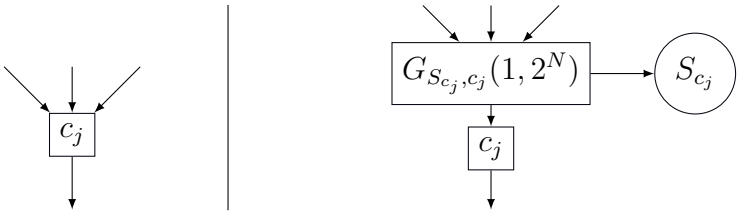


Figure 3.12: Insertion of gadget $G_{S_{c_j},c_j}(1, 2^N)$ to remove vertices v with $\sigma(v) < 0$ of our previous construction but keeping track of the fact that a particle has visited C_j or not.

Consider the previous reduction where we add a sink s_{C_j} to each clause vertex C_j and we add the gadget of paragraph 3.5.1 before C_j with $a = s_{C_j}$ and $b = C_j$ as illustrated in Figure 3.12.

In that case, let σ be the particle configuration on G_F such that for all vertices X_i, $\sigma(X_i) = \Gamma(x_i) + 2$ and for all other vertices v, $\sigma(v) = 0$ (see Figure 3.13). Let σ' be the particle configuration on G_F such that $\sigma'|_{V_0} = 0$, $\sigma'(S) = \Sigma_{i \leq n} \Gamma(x_i) - m$ and $\sigma'(s) = 1$ for all other sink vertex s.

Since the graph is acyclic, by replacing any vertex C_j with gadget $G_{S_{C_j},C_j}(1, 2^N)$ with $2^N \geq n \cdot \Sigma_{0 \leq i \leq n} \Gamma(x_i)$, we can ensure that the only particle that will be routed to vertex S_{C_j} in a maximal routing from σ is the first one arriving at C_j.

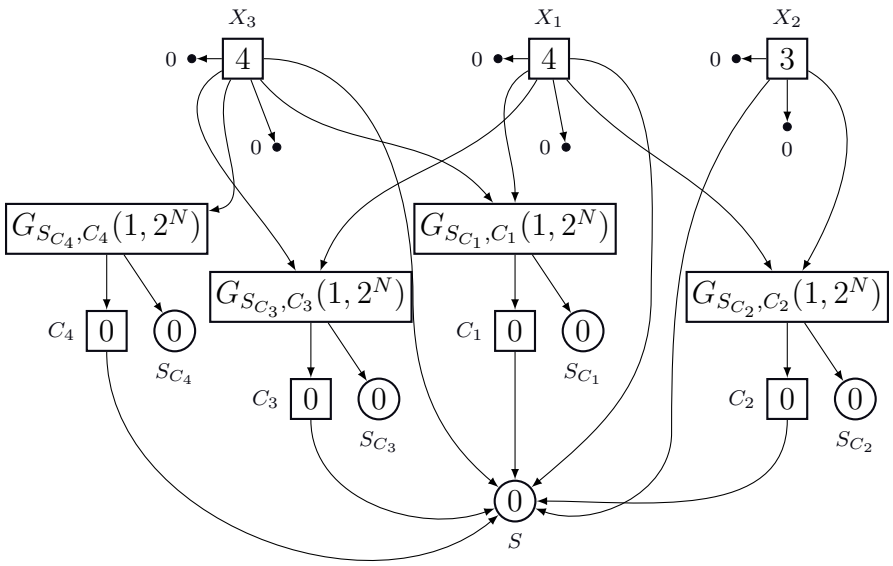


Figure 3.13: Reduction from $(*, \sigma) \to (*, \sigma')$ when $\sigma \geq 0$ and $\sigma'|_{V_0} = 0|_{V_0}$ to **MP-ARRIVAL**.

Then, the reduction follows from the same arguments than in the previous paragraph. This proves the following Lemma:

Lemma 3.5.2. *Problem $(*, \sigma) \to (*, \sigma')$ is **NP-Complete** on stopping graphs when $\sigma \geq 0$ and $\sigma'|_{V_0} = 0|_{V_0}$.*

Problem $(*, \sigma) \sim (*, \sigma')$ on stopping graphs

Now, we show that problem $(*, \sigma) \sim (*, \sigma')$ is also **NP-Complete**. By the same argument than for **MP-ARRIVAL** and **Linear-ARRIVAL**, we have the following lemma:

Lemma 3.5.3. *Problem $(*, \sigma) \sim (*, \sigma')$ is **NP-Complete** on stopping graphs when $\sigma \geq 0$ and $\sigma'|_{V_0} = 0|_{V_0}$.*

Proof. If $(*, \sigma) \to (*, \sigma')$, then clearly$(*, \sigma) \sim (*, \sigma')$. Conversely, since $\sigma \geq 0$, for all $\rho \in \mathcal{R}(\mathcal{G})$, there is a legal routing sequence from (ρ, σ) to a configuration (ρ', σ'') with $\sigma''|_{V_0} = 0|_{V_0}$ which implies by Lemma 2.3.12 that $\sigma' = \sigma''$. This shows the equivalence of problems $(*, \sigma) \sim (*, \sigma')$ and $(*, \sigma) \to (*, \sigma')$ when $\sigma \geq 0$ and $\sigma'|_{V_0} = 0|_{V_0}$ on stopping graphs. And, by Lemma 3.5.2 we have that $(*, \sigma) \sim (*, \sigma')$ is **NP-Complete** when $\sigma \geq 0$ and $\sigma'|_{V_0} = 0|_{V_0}$. □

Finally, since problem $(*, \sigma) \sim (*, \sigma')$ with $\sigma \geq 0$ and $\sigma'|_{V_0} = 0|_{V_0}$ is a subcase of problem $(*, \sigma) \sim (*, \sigma')$, we have the following theorem:

Lemma 3.5.4. *Problem $(*, \sigma) \sim (*, \sigma')$ is **NP-Complete** on stopping graphs.*

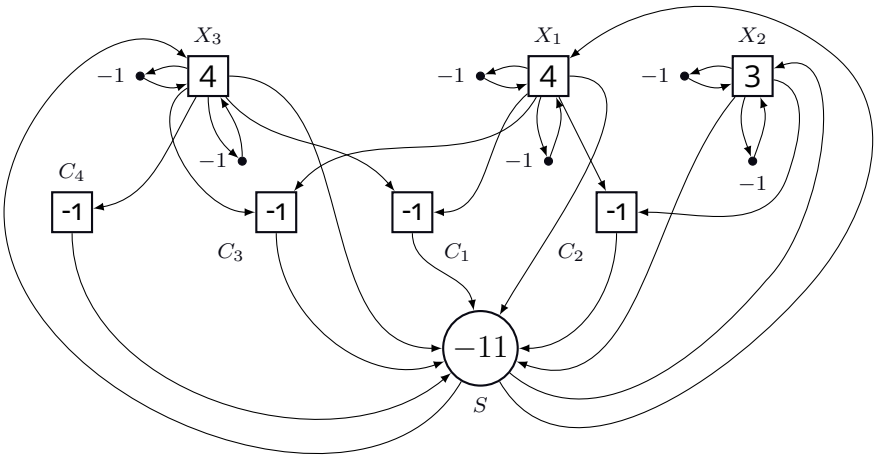


Figure 3.14: Construction of the strongly connected version of the graph G_F for the boolean formula $F = (x_1 \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2}) \wedge (x_1 \vee x_3) \wedge x_3$. Numbers depict the corresponding particle configuration σ.

3.5.2.2 Case where the graph is strongly connected

Now that we have proved that $(*, \sigma) \to (*, \sigma')$ and $(*, \sigma) \sim (*, \sigma')$ are both **NP-Complete** on stopping graphs, this subsection presents the proof of the following lemma:

Lemma 3.5.5. *Problem $(*, \sigma) \to (*, \sigma')$ is **NP-Complete** on strongly connected graphs.*

Consider an instance of problem $(*, \sigma) \to (*, \sigma')$ on a strongly connected graph G. We use a construction similar to the stopping case but for each X_i we add one arc from S to X_i in G_F. And we change σ such that $\sigma(S) = -\Sigma_{i \leq n}(\Gamma(x_i) + 2)$ and $\sigma(\textbf{true}_i) = \sigma(\textbf{false}_i) = -1$ for each $i \leq n$. This guarantees that when particles reach vertex S, vertex $false_i$ or vertex $true_i$, they cannot be routed anymore since there are $\Sigma_{i \leq n}(\Gamma(x_i) + 2)$ particles total in the system. We also change σ' such that $\sigma'(S) = -\Sigma_{i \leq n}(\Gamma(x_i) + 2) + \Sigma_{i \leq n}\Gamma(x_i) - m = -2 * n - m$ and $\sigma'(\textbf{true}_i) = \sigma'(\textbf{false}_i) = 0$.

Figure 3.14 is an example of construction of G_F and σ for the sinkless strongly connected case.

Since S also has to receive exactly $\Sigma_{i \leq n}\Gamma(x_i) - m$ particles, this case is thus equivalent to the stopping case. Hence, we have our result.

Theorem 3.5.6. *Problem $(*, \sigma) \sim (*, \sigma')$ is **NP-Complete** in the following classes of graphs:*

- *G is stopping (Lemma 3.5.4)*
- *G is stopping, $\sigma \geq 0$ and $\sigma'|_{V_0} = 0|_{V_0}$ (Lemma 3.5.3)*

*Problem $(*, \sigma) \to (*, \sigma')$ is **NP-Complete** in the following cases:*

- *G is stopping (Lemma 3.5.1)*
- *G is stopping, simple and acyclic (Lemma 3.5.1)*
- *G is stopping, $\sigma \geq 0$ and $\sigma'|_{V_0} = 0|_{V_0}$ (Lemma 3.5.2)*
- *G is strongly connected (Lemma 3.5.5)*

Restricted case of $(*, \sigma) \to (*, \sigma')$ on strongly connected graphs

Note that when σ' is positive and $|\sigma| = |\sigma'|$, by Lemma 2.4.7, it is clear that $(*, \sigma) \to (*, \sigma')$ always has a solution on strongly connected graphs.

3.6 . Problem $(\rho, \sigma) \xrightarrow{\sim} (*, \sigma')$

This section is dedicated to studying problems $(\rho, \sigma) \sim (*, \sigma')$ and $(\rho, \sigma) \to (*, \sigma')$. We demonstrate that these problems are not only equivalent to each other but also equivalent to MP-ARRIVAL.

In the first part of this section, we establish the equivalence between the linear version of the problem $(\rho, \sigma) \sim (*, \sigma')$ and MP-ARRIVAL. We demonstrate that these two problems are equivalent, meaning that a solution to one problem can be directly translated into a solution for the other problem.

In the second part of this section, we focus on establishing the equivalence between $(\rho, \sigma) \to (*, \sigma')$ and MP-ARRIVAL.

To prove these equivalences, we examine several subcases of these problems as outlined in Figure 3.15.

3.6.1 . Equivalence between $(\rho, \sigma) \sim (*, \sigma')$ and MP-ARRIVAL

Lemma 3.6.1. *Problems $(\rho, \sigma) \sim (*, \sigma')$ and **Linear ARRIVAL** are equivalent on stopping rotor graphs.*

Proof. Since there is no restriction on the routing sequence, we have $(\rho, \sigma) \sim (*, \sigma') \Leftrightarrow (\rho, \sigma - \sigma') \sim (*, 0_V)$.

Now, consider a configuration $\sigma^+ \sim 0_V$ such that $\sigma^+(v) + \sigma(v) - \sigma'(v) \geq 0$ for all $v \in V_0$.

We show that such a configuration always exists by using the same construction than in the proof of Lemma 2.5.18. Namely, consider the strongly connected graph $G' = (V_0' \cup S_0', \mathcal{A}', h, t, \theta)$ such that $V_0' = V_0 \cup S_0$, $S_0' = \emptyset$ and $\mathcal{A}' = \mathcal{A} \cup \mathcal{A}_{S_0}$ with $\mathcal{A}_{S_0}$ being a set with exactly one arc from each sink in S_0 to each vertex of V_0. Let $\rho_{G'}$ be the rotor configuration such that for any vertex

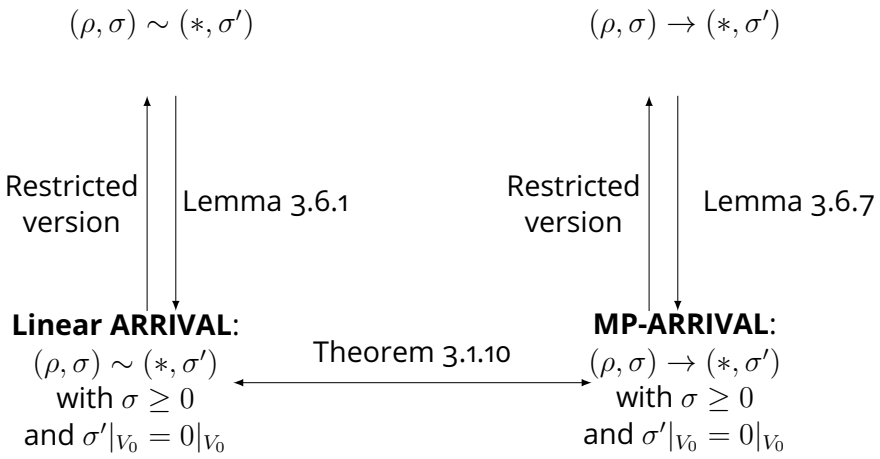


Figure 3.15: Equivalence and Complexity Results for problem $(\rho, \sigma) \xrightarrow{\sim} (*, \sigma')$. An arrow represents a polynomial reduction from the problem at the tail to the problem at the head.

$v \in V_0$ we have $\rho_{G'}(v) = \rho(v)$ and for any other vertex $s \in V_0' \setminus V_0$, $\rho_G'(s)$ is any outgoing arc of s. Let p be a positive period vector of G'.

Consider the firing sequence from $(\rho, 0|_{V_0'})$ to some (ρ, σ^+) with $\sigma^+ \in \mathcal{P}_{V'}(G')$ such that each vertex $s \in S_0$ is positively fired exactly $p(s)$ times in r and none of the other vertices are fired. Consider graph G once again. By construction, for any $v \in V_0$, we have $\sigma^+(v) > 0$ because p is positive. Hence, by choosing p big enough, we can ensure that $\sigma^+(v) + \sigma(v) - \sigma'(v) \geq 0$ for all $v \in V_0$.

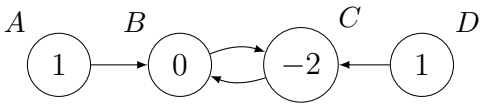
Thus, we have $(\rho, \sigma - \sigma') \sim (*, 0_V) \Leftrightarrow (\rho, \sigma - \sigma' + \sigma^+) \sim (*, 0_V)$, but problem $(\rho, \sigma - \sigma' + \sigma^+) \sim (*, \sigma^+) \sim (*, 0_V)$ is an instance of $(\rho, \sigma) \sim (\rho', \sigma')$ with $\sigma|_{V_0} \geq 0$ and $\sigma'|_{V_0} = 0_{V_0}$ which is exactly **Linear-ARRIVAL**. □

Corollary 3.6.2. *Problems $(\rho, \sigma) \sim (*, \sigma')$ and **MP-ARRIVAL** are equivalent.*

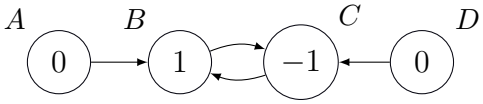
Proof. By Lemma 3.6.1, we have $(\rho, \sigma) \sim (*, \sigma') \Leftrightarrow$ **Linear-ARRIVAL**. And by Theorem 3.1.10, we have that **Linear ARRIVAL** and **MP-ARRIVAL** are equivalent, hence we have our result. □

3.6.2 . Equivalence between $(\rho, \sigma) \to (*, \sigma')$ and MP-ARRIVAL

At first glance, **MP-ARRIVAL** and $(\rho, \sigma) \to (*, \sigma')$ may seem very similar. However, there is a key difference between them: in **MP-ARRIVAL** we proceed all legal routings whereas in $(\rho, \sigma) \to (*, \sigma')$ we may have to avoid proceeding some legal routings in order to reach σ'. This is illustrated in Example 3.16.



(a) In this example there is only one rotor configuration ρ, and we choose σ such that $\sigma(A) = 1, \sigma(B) = 0, \sigma(C) = -2$ and $\sigma(D) = 1$.



(b) We depict the rotor particle configuration (ρ', σ') with $\rho' = \rho$ and $\sigma'(A) = 0, \sigma'(B) = 1, \sigma'(C) = -1$ and $\sigma'(D) = 0$.

Figure 3.16: Even if proceeding successive routings on vertices A, B then D is a legal routing sequence, configuration (ρ', σ') will never be reached by such a sequence. However, it suffices to route successively A and D in any order to reach configuration (ρ', σ').

But, we can restrict ourselves to some routing sequences that avoid proceeding too many routings to reach σ': σ'-routing sequences.

Definition 3.6.3 (σ'-routing sequence)**.** *Let $\rho_0 \in \mathcal{R}(G)$ and let σ' be a particle configuration on a graph G, a* σ'-routing sequence *is a legal sequence of rotor particle configurations $(\rho_0, \sigma_0), (\rho_1, \sigma_1), \cdots, (\rho_n, \sigma_n)$ such that for any $i \leq n$ there exists a vertex $v \in V$ such that $min(\sigma_i(v) - \sigma'(v), \sigma_i(v)) \geq 1$ and $(\rho_{i+1}, \sigma_{i+1}) = routing_v^+(\rho_i, \sigma_i)$. We denote the fact that (ρ_n, σ_n) is obtained from (ρ_0, σ_0) by a σ'-routing sequence with $(\rho_0, \sigma_0) \xrightarrow{\sigma'} (\rho_n, \sigma_n)$.*

Proposition 3.6.4 (Existence of σ'-routing sequence)**.** *Given a rotor particle configuration (ρ, σ) and a particle configuration σ',*

$$\exists \rho' \in \mathcal{R}(G), (\rho, \sigma) \to (\rho', \sigma') \Leftrightarrow \exists \rho'' \in \mathcal{R}(G), (\rho, \sigma) \xrightarrow{\sigma'} (\rho'', \sigma')$$

and all vertices $v \in V$ are routed less times in $(\rho, \sigma) \xrightarrow{\sigma'} (*, \sigma')$ than in $(\rho, \sigma) \to (*, \sigma')$.

Proof. It is pretty clear that if $(\rho, \sigma) \xrightarrow{\sigma'} (*, \sigma')$ then $(\rho, \sigma) \to (*, \sigma')$. Now, we prove the other implication. Assume that $(\rho, \sigma) \xrightarrow{r} (*, \sigma')$, with r being a routing sequence with routing vector R. Now, consider $r_{\sigma'}$ a maximal r-bounded σ'-routing sequence starting from (ρ, σ) with routing vector R_σ. This is a maximal legal routing sequence such that for all $v \in V$ we have $R_\sigma(v) \leq R(v)$. Let (ρ_1, σ_1) be the rotor particle configuration obtained by applying $r_{\sigma'}$ to (ρ, σ). Since $r_{\sigma'}$ is maximal, there is no more σ'-routing doable at (ρ_1, σ_1). Assume that $\sigma_1 \neq \sigma'$, then, since $|\sigma_1| = |\sigma'|$ there exists a vertex v such that $\sigma_1(v) > \sigma'(v)$.

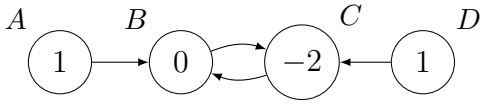
Suppose that $v \in S_0$, since sink vertices cannot be routed, this means that there exists some ingoing neighbour of v that has been routed more times in $r_{\sigma'}$ than in r which is a contradiction with the fact that $r_{\sigma'}$ is r-bounded.

Consider now that $v \in V_0$. Since $\sigma_1(v) > \sigma'(v)$ and there is no σ'-routing doable at v, we have either $R_\sigma(v) = R(v)$ or $\sigma_1(v) \leq 0$.

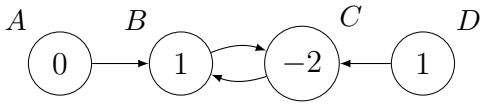
In the first case, remark that v can only gain particles when other vertices are routed. Hence, since $\sigma_1(v) > \sigma'(v)$, completing R_σ up to R can only increase the number of particles on v and it will not be equal to $\sigma'(v)$ after proceeding R which is a contradiction.

In the second case, since $\sigma_1(v) \leq 0$ and $\sigma_1(v) > \sigma'(v)$, then, as long as $\sigma_1(v) < 0$, completing R_σ up to R can only increase the number of particles on v and if at some point $\sigma_1(v) \geq 0$, the number of particles on v may decrease but never below zero. Hence it will not be equal to $\sigma'(v)$ after proceeding R which is a contradiction. This also proves that $R_\sigma \leq R$. □

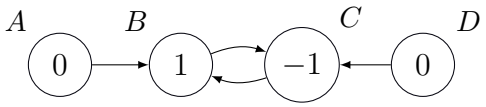
An example of σ'-routing sequence is depicted on Figure 3.17.



(a) Let σ' be the particle configuration $\sigma(A) = 0, \sigma(B) = 1, \sigma(C) = -1$ and $\sigma(D) = 0$. In the configuration depicted in this example, there are two legal σ'-routings doables, one at A and one at D.



(b) Rotor particle configuration obtained after proceeding a σ'-routing at A. In this configuration, there is an unique legal σ'-routing doable which is at D.



(c) Rotor particle configuration obtained by proceeding a σ'-routing at D. From here there is no more legal σ'-routing.

Figure 3.17: Presentation of a σ'-routing sequence on the example of Figure 3.16.

Corollary 3.6.5 (Maximal σ'-routing is unique)**.** *Let R_σ be the routing vector of a maximal σ'-routing sequence $(\rho, \sigma) \xrightarrow{R_\sigma} (\rho_1, \sigma_1)$ and R'_σ be the routing vector of a maximal σ'-routing sequence $(\rho, \sigma) \xrightarrow{R'_\sigma} (\rho_2, \sigma_2)$. Then $(\rho_1, \sigma_1) = (\rho_2, \sigma_2)$ and $R_\sigma = R'_\sigma$.*

Now we have all necessary results to show the following theorem:

Theorem 3.6.6. *Problem $(\rho, \sigma) \to (*, \sigma')$ is equivalent to **MP-ARRIVAL**.*

3.6.2.1 Reduction of $(\rho, \sigma) \to (*, \sigma')$ to MP-ARRIVAL

Consider an instance of the reachability problem : $(\rho, \sigma) \to (*, \sigma')$. Let $t(v) = \sigma'(v) - \sigma(v)$. We first construct a new graph G' from G by adding a sink s_v for

each vertex $v \in V_0$ such that $\sigma(v) < 0$ and for each vertex v with $t(v) > 0$ and $\sigma(v) \geq 0$. Then, for each vertex v :

- if $\sigma(v) < 0$, add a gadget $G_{s_v,v}(|\sigma(v)|, 2^N)$ before v as illustrated in Figure 3.12.
- if $t(v) > 0$ and $\sigma(v) \geq 0$, add a gadget $G_{s_v,v}(t(v), 2^N)$ before v.

Let σ_0 be a particle configuration such that for all $v \in V_0$ with $\sigma(v) < 0$, $\sigma_0(v) = 0$, for all other vertices $w \in V$ we have $\sigma_0(w) = \sigma(w)$, and for all gadgets $\sigma_0(G_{s_v,v}) = 0$. Let σ_S be the particle configuration on G' such that for all $s_v \in S'_0$ we have $\sigma_S(s_v) = \sigma'(v) - \sigma(v)$ and for all $v \in V'_0$ we have $\sigma_S(v) = 0$.

This construction is illustrated on Figure 3.18 for the example of Figure 3.16.

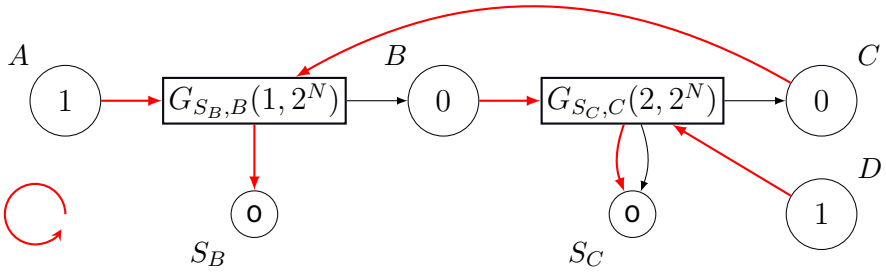


Figure 3.18: Reduction from the instance of problem $(\rho, \sigma) \to (*, \sigma')$ presented on Figure 3.16 to an instance of **MP-ARRIVAL** on a graph G'.

The main idea is that each sink vertex s_v captures the right amount of particles that should be on v at the end of the routing. This transformation works well unless a gadget $G_{s_v,v}$ is routed too many times and the vertex s_v receives more than $\sigma_S(s_v)$ particles. By Corollary 3.1.6, the number of routings at v (hence at $G_{s_v,v}$) cannot be higher than $b = |\sigma| \cdot \mathcal{MAX}_{v \in V}(|\mathcal{A}^+(v)|) \cdot e^{|\mathcal{A}|/e}$. Hence, by choosing $N > \log(b)$ for our gadgets, we ensure that there cannot be more than $\sigma_S(s_v)$ particles routed up to sink s_v.

Lemma 3.6.7. *There is a polynomial reduction from problem $(\rho, \sigma) \to (*, \sigma')$ to **MP-ARRIVAL** on stopping graphs.*

3.6.2.2 Problem $(\rho, \sigma) \to (*, \sigma')$ on strongly connected graphs

Lemma 3.6.8. *Problem $(\rho, \sigma) \to (*, \sigma')$ on strongly connected graph is equivalent to problem $(\rho, \sigma) \to (*, \sigma')$ on stopping graphs.*

Proof. First, consider an instance of problem $(\rho, \sigma) \to (*, \sigma')$ on a stopping graph G. Let σ_{sc} be the particle configuration such that $\sigma_{sc}|_{V_0} = \sigma|_{V_0}$ and such that for all sink vertex $s \in S_0$ we have $\sigma_{sc}(s) = \sigma(s) - k$ with $k = |V_0| \cdot$

$\mathcal{MAX}_{v\in V}|\sigma(v)|$. Let σ'_{sc} be the particle configuration such that $\sigma'_{sc}|_{V_0} = \sigma'|_{V_0}$ and such that for all sink vertex $s \in S_0$ we have $\sigma'_{sc}(s) = \sigma'(s) - k$. Then let $G' = (V'_0 \cup S'_0, \mathcal{A}')$ be the strongly connected graph where we add one outgoing arc to each sink vertex towards any other vertex. Since there are not enough particles in the system to make sinks positive, they are never routed in both G and G'. So, $(\rho, \sigma) \to (*, \sigma')$ on G if and only if $(\rho, \sigma_{sc}) \to (*, \sigma'_{sc})$.

Now, consider an instance of problem $(\rho, \sigma) \to (*, \sigma')$ on a strongly connected graph G. Then, we can use the same construction than in paragraph 3.6.2.1 to construct a stopping graph G' such that there is a solution on G' if and only if there is a solution on G.

□

Then by Lemma 3.6.6 we have the following Lemma.

Lemma 3.6.9. *Problem $(\rho, \sigma) \to (*, \sigma')$ on strongly connected graph is equivalent to **MP-ARRIVAL**.*

Our results of this section can be summarized by the following theorem.

Theorem 3.6.10. *Problem $(\rho, \sigma) \sim (*, \sigma')$ is equivalent to **MP-ARRIVAL** in the following classes of graphs:*

- *G is stopping (Corollary 3.6.2);*
- *G is stopping, $\sigma \geq 0$ and $\sigma'|_{V_0} = 0|_{V_0}$ which is exactly problem **Linear ARRIVAL** (Theorem 3.1.10).*

*Problem $(\rho, \sigma) \to (*, \sigma')$ is equivalent to **MP-ARRIVAL** in the following cases:*

- *G is stopping (Lemma 3.6.6);*
- *G is stopping, $\sigma \geq 0$ and $\sigma'|_{V_0} = 0|_{V_0}$ which is exactly **MP-ARRIVAL**;*
- *G is strongly connected (Lemma 3.6.9).*

3.7 . Problem $(*, \sigma) \xrightarrow{\sim} (\rho', \sigma')$

This section provides a summary of our results on specific subcases of the problem $(*, \sigma) \xrightarrow{\sim} (\rho', \sigma')$. Although we have already established some reductions for general cases, further properties are needed to determine the complexity of the problems $(*, \sigma) \to (\rho', \sigma')$ on stopping graphs and strongly connected graphs. We are actively working on this and plan to publish detailed results in the near future.

3.7.1 . Problem $(*, \sigma) \sim (\rho', \sigma')$

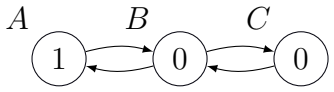
Lemma 3.7.1. *Problem $(*, \sigma) \sim (\rho', \sigma')$ is equivalent to **MP-ARRIVAL**.*

Proof. Since $(*, \sigma) \sim (\rho', \sigma')$ and $(\rho, \sigma) \sim (*, \sigma')$ are equivalent by Lemma 3.4.1 and since $(\rho, \sigma) \sim (*, \sigma')$ and **MP-ARRIVAL** are equivalent by Corollary 3.6.2, we have that $(*, \sigma) \sim (\rho', \sigma')$ is equivalent to **MP-ARRIVAL**. □

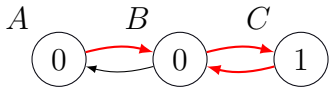
3.7.2 . Problem $(*, \sigma) \to (\rho', \sigma')$

Restricted case of $(*, \sigma) \to (\rho', \sigma')$ on strongly connected graph

Sadly, the restricted case of $(*, \sigma) \to (\rho', \sigma')$ when σ' is positive, $|\sigma| = |\sigma'|$ and the graph is strongly connected cannot be solved as easily as for problems $(\rho, \sigma) \to (*, \sigma')$. Figure 3.19 shows an example for which there is no solution for problem $(\rho, \sigma) \to (*, \sigma')$ on a strongly connected graph.



(a) Particle configuration σ



(b) Rotor particle configuration (ρ', σ')

Figure 3.19: Instance of problem $(*, \sigma) \to (\rho', \sigma')$ with $\sigma' > 0$ and $|\sigma| = |\sigma'|$ for which there is no rotor configuration ρ such that $(\rho, \sigma) \to (\rho', \sigma')$. Indeed, since the last movement of the particle before reaching C necessarily is a positive routing at B, the ending rotor configuration at B has to be (B, A).

But, in the case where the graph is sinkless and the rotor particle configuration (ρ', σ') is recurrent, we have the following property.

Lemma 3.7.2. *Given a particle configuration σ and recurrent rotor particle configuration (ρ', σ') such that $|\sigma| = |\sigma'|$, the problem $(*, \sigma) \to (\rho', \sigma')$ always has a solution.*

Proof. By the general case for the same problem, we have that there always exists a rotor configuration ρ such that there is a routing vector (non-necessarily legal) that transforms (ρ, σ) in (ρ', σ'). We have $(\rho, \sigma) \sim (\rho', \sigma')$. Since (ρ', σ') is recurrent, we also have $|\sigma'| > 0$ and $|\sigma| > 0$. Thus, by proceeding a long enough legal routing sequence from (ρ, σ), we reach some rotor particle configuration (ρ^*, σ^*) that is recurrent. And since there exists a legal routing sequence between all recurrent rotor particle configurations within the same equivalence class (Corollary 2.5.16). Then, there is a legal routing sequence from (ρ, σ) to (ρ', σ').

□

3.7.2.0.1 Problem $(*, \sigma) \to (\rho', \sigma')$ with SP-configurations on stopping graphs

Recall that problem $(\rho, \sigma) \to (*, \sigma')$ on a stopping graph with σ being a SP-configuration and σ' being such that $\sigma'|_{V_0} = 0|_{V_0}$ is exactly **SP-ARRIVAL**, hence,

it belongs to **NP** ∩ **co-NP**. Here, we prove that the "reverse" problem is easy.

Proposition 3.7.3. *Given a SP-configuration σ and a particle configuration σ' such that $\sigma'|_{V_0} = 0_{V_0}$, there exists a solution to problem $(*, \sigma) \to (\rho', \sigma')$ if and only if there is a directed path in $G(\rho'^{-})$ from w to any vertex in S_0.*

Proof. First assume there exists a directed path from w to a vertex $s \in S_0$ in $G(\rho'^{-})$. We can then construct a legal routing sequence as follows: let ρ be the configuration where $\rho(v) = \theta_v^-(\rho'(v))$ for all vertices v in the path from w to s, and $\rho(v') = \rho'(v')$ for all other vertices v'. Then, we can route positively the vertices in the path from w to s one after another, which may lead the particle to the sink.

Conversely, suppose there is a legal routing sequence from $(*, \sigma)$ to (ρ', σ'). This means that the particle may reach a sink s, so the last vertex the particle visits before reaching s must be in V_0. Moreover, the particle can only visit vertices v such that there is a directed path from v to s in $G(\rho'^{-})$. Therefore, there is a directed path from w to this last vertex in $G(\rho'^{-})$, and by transitivity of the directed edges, there is a directed path from w to a vertex in S_0.

□

3.8 . Legal problems with non-fixed particle configuration(s)

In this section, we will discuss the common results for three problems: $(\rho, *) \to (\rho', \sigma')$, $(\rho, \sigma) \to (\rho', *)$ and $(\rho, *) \to (\rho', *)$. These problems share the property that at least one of the particle configurations σ or σ' is not known in advance.

Always solvable cases

It is worth noting that the problem $(\rho, *) \to (\rho', *)$ always has a solution, both for stopping and strongly connected graphs. This is because we can use the same argument as in the general case. Namely, we can choose a particle configuration σ such that $\sigma(v) \geq |\mathcal{A}^+(v)| - 1$ for all $v \in V_0$, which guarantees the existence of a legal routing sequence. Then, we can align the initial and final configurations ρ and ρ' using the positive aligning routing vector $R^+(\rho, \rho')$.

Furthermore, this also implies that for problem $(\rho, \sigma) \to (\rho', *)$, if σ is such that $\sigma(v) \geq |\mathcal{A}^+(v)| - 1$ for all $v \in V_0$, then there always exists a legal routing sequence from (ρ, σ) to $(\rho', *)$.

Note that there is no similar result for problem $(\rho, *) \to (\rho', \sigma')$. However, we have some results for this problem in the particular case where (ρ', σ') is a recurrent rotor particle configuration.

Case of a recurrent rotor particle configuration

Proposition 3.8.1. *Given (ρ', σ') a recurrent rotor-particle configuration (Definition 2.5.15) on a strongly connected graph with a particle configuration σ', then there is a particle configuration σ such that $(\rho, \sigma) \to (\rho', \sigma')$ is legal.*

Proof. Consider the aligning vector R that transforms ρ into ρ'. Then, we choose a particle configuration σ_0 such that $\sigma_0(v) = R(v)$ for all $v \in V_0$. Next, observe that there exists a legal routing sequence from (ρ, σ_0) to some (ρ', σ'_0) using the routing vector R. As a result, we have $(\rho, \sigma_0 - \sigma'_0) \overset{R}{\sim} (\rho', 0)$. Therefore, we also have $(\rho, \sigma_0 - \sigma'_0 + \sigma') \overset{R}{\sim} (\rho', \sigma')$. Since (ρ', σ') is recurrent, then, by Corollary 2.5.16, we have constructed a legal routing sequence from $(\rho, \sigma_0 - \sigma'_0 + \sigma')$ to (ρ', σ'). □

While we have obtained some initial results in studying these problems, there is still much more to explore and investigate.

Future Work

This chapter established the complexity of several reachability problems within our rotor-routing framework (see Table 3.2). In particular, we were able to demonstrate that **MP-ARRIVAL** is equivalent to the reachability problem $(\rho, \sigma) \to (*, \sigma')$. And this result allowed us to prove that $(\rho, \sigma) \to (*, \sigma')$ and $(\rho, \sigma) \sim (*, \sigma')$ are equivalent. This finding is encouraging because it indicates that the complexity of **MP-ARRIVAL** does not come from the legality constraint but is inherent to the rotor-routing mechanism and the graph topology. This reinforces our approach of studying **MP-ARRIVAL** on specific graph topologies to enhance our understanding of the problem. Recently, we solved **MP-ARRIVAL** for Path Multigraphs, which has been published in [6]. This idea can be pursued for other problems in this chapter. Exhibiting reductions to restricted versions of them can provide insights into their complexity and identify key factors that make them challenging. We have an element of answer for problem $(*, \sigma) \to (*, \sigma')$ which is **NP-Complete** even for simple acyclic stopping graphs.

Since the goal of this chapter was also to gain a better understanding of **ARRIVAL**, we summarized all results of this chapter concerning **ARRIVAL** in Table 3.3.

While we have obtained complexity results for some problems of Table 3.2, or at least for some subcases, there are still general problems for which we cannot give a complexity result yet, notably the "dual" version of **MP-ARRIVAL** $(*, \sigma) \to (\rho', \sigma')$. We conjecture here that $(*, \sigma) \to (\rho', \sigma')$ is also equivalent to **MP-ARRIVAL**.

We also demonstrated that certain problems listed in Table 3.2 always have a solution. However, we did not delve into the fact that finding this solution can be complex and may even require exponential time to compute.

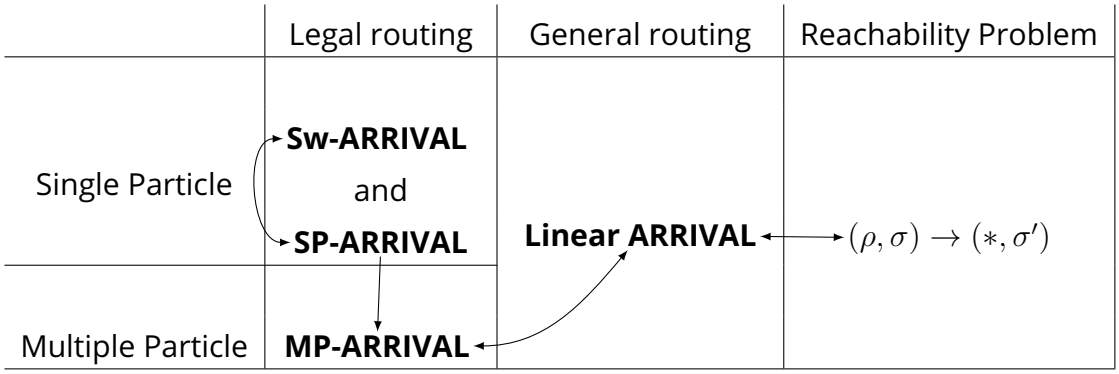


Table 3.3: Relations between problems of **ARRIVAL** and the reachability problem $(\rho, \sigma) \to (*, \sigma')$. Arrows depict direct reductions between problems (e.g., **SP-ARRIVAL** is a restriction of **MP-ARRIVAL** to instances with a single particle). These reductions also proved that all problems in this table belong to **NP** ∩ **co-NP** since **Linear ARRIVAL** belongs to **NP** ∩ **co-NP**.

Finally, as an open problem, it is possible to study the equivalence class versions of these problems. As an example, consider the following problem: given $[\rho_1]$ a rotor class, $[\sigma_1]$ and $[\sigma_2]$, two particle classes, does there exist $\rho \in [\rho_1]$, $\sigma \in [\sigma_1]$ and $\sigma' \in [\sigma_2]$ such that for some $\rho' \in \mathcal{R}(G)$ we have $(\rho, \sigma) \to (\rho', \sigma')$.

4 - SP-ARRIVAL on Treelike-Multigraphs

All results presented in this chapter have been published in [3]. We only consider legal routing sequences, stopping graphs with $S_0 \neq \emptyset$ and SP-configurations, namely configurations such that for all vertices $v \in V$, we have $\sigma(v) = 0$ except for one vertex $w \in V$ for which $\sigma(w) = 1$. Note that in this context of legality, we simplify the writing of operation $\mathbf{routing}^+$ by $\mathbf{routing}$ as there is no risk of confusion. In the same way, a positive cycle push will be denoted by a *cycle push* as there is no risk of confusion either.

We recall the following definition of a rotor SP-configuration. Let $u \in V$. We denote by (ρ, u) the rotor SP-configuration such that the rotor configuration on the graph is ρ and the particle configuration is such that the single vertex with a particle on it is u.

In this context we give an alternative definition of the rotor walk defined in Chapter 2. This definition is commonly used in the literature when dealing with SP-configurations.

Definition 4.0.1 (Rotor Walk)**.** *A* rotor walk *is a (finite or infinite) sequence of rotor SP-configurations $(\rho_i, u_i)_{i\geq 0}$, which is recursively defined by $(\rho_{i+1}, u_{i+1}) = \mathbf{routing}(\rho_i, u_i)$ with $u_i \in V_0$ being the only vertex with a particle on it at step i.*

The properties and definitions presented below are restatements from Chapter 2, which have been adapted to the context of this chapter.

The following lemma is a classical result on rotor walks (cf Lemma 16 in [41]) and is a reformulation of Lemma 2.5.20 adapted to the context of this chapter.

Lemma 4.0.2 (Finite number of steps (Lemma 16 in [41]))**.** *If G is stopping then any maximal rotor walk in G is finite.*

The main objective of this chapter is to study the sink that will be reached by a maximal rotor walk from a SP-configuration, if the rotor walk is finite.

Definition 4.0.3 (Exit Sink)**.** *Let $u \in V$, let ρ be a rotor configuration, if the maximal rotor walk starting from (ρ, u) is finite in G, then the sink reached by such a rotor walk is denoted by $\mathbf{S}_G(\rho, u)$ and called* exit sink *of u for the rotor configuration ρ in G.*

Definition 4.0.4 (Exit Pattern)**.** *For a rotor configuration ρ on a stopping rotor graph G, the* exit pattern *is the mapping that associates with each vertex $u \in V$, its exit sink $\mathbf{S}_G(\rho, u)$.*

4.1 . SP-ARRIVAL and Complexity Issues

With our notations, **SP-ARRIVAL** (see [31]) can be expressed as the following decision problem:

In a stopping rotor graph G, given a vertex $u \in V$, and $s \in S_0$ does $\mathbf{S}_G(\rho, u) = s$?

Problem **SP-ARRIVAL** belongs to the complexity class **NP** $\cap$ **co-NP**, as demonstrated in Section 3.

4.1.1 . Cycle Pushing

In order to speed-up the rotor walk process, we use the positive cycle push operation to avoid computing every step of the walk.

We recall the following lemma that we have already stated in Chapter 2.

Lemma 4.1.1 (Finite Number of Cycle Pushes)**.** *Given a stopping rotor graph, any sequence of cycle pushes is finite.*

The previous result is well known in rotor walk studies ([50]). It implies that, by processing a long enough sequence of successive positive cycle pushes, the resulting configuration contains no directed cycles, i.e., we reach the only acyclic configuration within the same rotor class as the initial rotor configuration (Proposition 2.3.7). Such a sequence of cycle pushes is called *maximal*.

The following result can be found in [41] but it is also a direct consequence of Proposition 2.4.9.

Lemma 4.1.2 (Exit Pattern conservation for Cycle Push ([41]))**.** *If G is a stopping rotor graph, for any rotor configuration ρ and configuration ρ' obtained from ρ by a cycle push, the exit pattern for ρ and ρ' is the same.*

Recall that by Proposition 2.3.7, the rotor configuration obtained by a maximal cycle push sequence is unique.

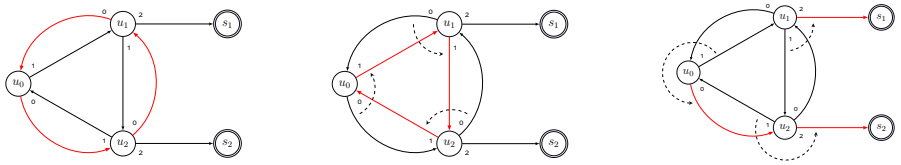
Definition 4.1.3 (Destination Forest)**.** *We call the rotor configuration obtained by a maximal cycle push sequence on ρ the* Destination Forest *of ρ, denoted by $D(\rho)$ as it does not depend on the starting particle configuration.*

The destination forest has a simple interpretation in terms of rotor walks: start a rotor walk by putting a particle on any vertex of a stopping graph G; consider a vertex $u \in V_0$; if the particle ever reaches u, it will leave u by arc $D(\rho)(u)$ on the last time it enters u.

In an acyclic configuration like $D(\rho)$, finding the exit pattern is very simple. We use the same kind of argument as for the subcase of problem $(*, \sigma) \to (\rho', \sigma')$ where σ is a SP-configuration, precisely:

Lemma 4.1.4 (Path to a sink)**.** *If there is a directed path between $u \in V$ and $s \in S_0$ in $G(\rho)$ then $\mathbf{S}_G(\rho, u) = s$. It follows that from $D(\rho)$ one can compute the exit pattern of ρ in time complexity $O(|V|)$.*

This gives us a new approach since computing the exit pattern of a configuration ρ can be done by computing its Destination Forest $D(\rho)$. Observe that by computing the destination Forest, we are solving a problem harder than **SP-ARRIVAL** because we compute the exit sink of all vertices simultaneously.



(a) Red Rotor Configuration ρ in red lines with rotor order on each vertex described by increasing numbers.

(b) The red configuration in dashes is obtained from a Cycle Push on the cycle $\{(u_0,u_2),(u_2,u_1),(u_1,u_0)\}$.

(c) Destination Forest (depicted by red arcs in dashes) computed by two successive cycle pushes on ρ.

Figure 4.1: Computation of the Destination Forest by successive cycle push operations.

Remark. *Pushing a cycle of length k is a "shortcut" in a rotor walk as it allows only one operation to simulate k steps of the rotor walk. However, the strategy consisting of pushing cycles until the Destination Forest is reached (which is always the case if the graph is stopping) can still take an exponential time. Indeed, let us consider the instance depicted on Figure 3.2 again. As each cycle in $G(\rho)$ is at most of size two, we are at most dividing by two the number of steps used in the maximal rotor walk.*

So, processing successive cycle pushes is not a suitable strategy to solve **SP-ARRIVAL** efficiently in general. However, the notion of equivalence classes that we stated in Chapter 2 is based on cycle pushing. So we state a characterization of a rotor configuration equivalence class in this chapter's context.

Lemma 4.1.5 (Equivalence Class [41])**.** *Two rotor configurations ρ_1 and ρ_2 are in the same equivalence class if they have the same Destination Forest.*

This is a direct consequence of Proposition 2.3.7.

The following lemma was stated in [41], under the name Corollary 14. It shows that equivalence of configurations is preserved by the routing of a particle to a sink.

Lemma 4.1.6 (Corollary 14 in [41])**.** *Let ρ_1, ρ_2 be two equivalent configurations. Let ρ'_1 and ρ'_2 be the rotor configurations obtained respectively from (ρ_1, u) and (ρ_2, u) by maximal rotor walk with u being a vertex of V. Then ρ'_1 and ρ'_2 are equivalent.*

Proof. Since $\rho_1 \sim \rho_2$, there exists a cycle push sequence from ρ_1 to ρ_2 and by Proposition 2.4.9, we then have that the loop-erased paths of the maximal rotor walk from (ρ_1, u) and (ρ_2, u) are the same. Hence, ρ_1' and ρ_2' are the same on this path and the only difference may be circuits that were already existing in ρ_1 and ρ_2, but since $\rho_1 \sim \rho_2$, ρ_2 those cycles can be obtained from one another by a cycle push sequence, thus we have that $\rho_1' \sim \rho_2'$. □

4.2 . Simple Path Graph

In this section we give some results on a particular class of graphs in order to give intuition behind our following work on trees. The simple path graph on $n+2$ vertices is defined by $V = \{s_0, u_1, u_2, \ldots, u_n, s_1\}$ and each u_i for $i \in \{1, .., n\}$ has an arc going to the vertex on the left and another one going to the vertex on the right. Define then $V_0 = \{u_1, u_2, ..., u_n\}$ together with $S_0 = \{s_0, s_1\}$.

See Figure 4.2 for the simple path graph with $n = 4$.

In this section we will not only show that **SP-ARRIVAL** can be solved in linear time on simple path graphs but that we can solve **MP-ARRIVAL** by calculating the exit pattern for ρ in linear time, *i.e.*, solving **SP-ARRIVAL** simultaneously for multiple particles.

However, note that the maximum number of steps of a rotor walk (or a cycle push sequence) in a simple path graph (as the one presented at the beginning of Chapter 1) is not exponential but quadratic in n. Indeed, one can check that the starting configuration that maximizes this number of steps is the configuration where all arcs are directed towards the central vertex $u_{\lceil n/2 \rceil}$ (which is also the starting vertex in the case of the rotor walk).

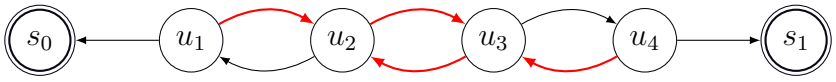
4.2.1 . Routing One Particle on a Path Graph

It is an easy observation that the only elementary cycles in this graph are of length 2, and that pushing such a cycle in a given configuration does not change the global numbers of arcs respectively directed towards s_0 and s_1 in the configuration. Hence the following definition:

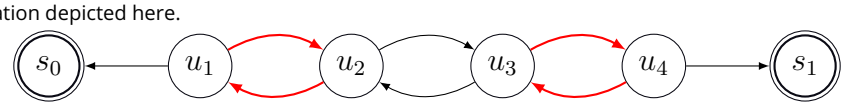
Definition 4.2.1. *We say that a vertex $u_i \in V_0$ is* directed towards s_0 *if $h(\rho(u_i)) \in \{u_{i-1}, s_0\}$. Otherwise, u_i is directed towards s_1 (see Figure 4.2a). We denote by $n_1(\rho)$ the number of vertices of V_0 that are directed towards s_1.*

From the observation above, it will follow that we can compute the exit pattern of ρ without having to process the sequence of cycle pushes, relying only on the computation of $n_1(\rho)$. To do so, we characterise the equivalence of rotor configurations that applies only in the case of a path graph.

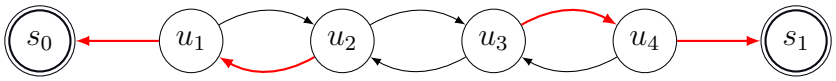
Lemma 4.2.2 (Exit Sink Characterization)**.** *Two configurations are equivalent if and only if they have the same number n_1. In particular, for any configuration ρ, if $1 \leq i \leq n - n_1(\rho)$, then $\mathbf{S}_G(\rho, u_i) = s_0$, otherwise $\mathbf{S}_G(\rho, u_i) = s_1$.*



(a) u_1 and u_2 are directed towards s_1, and u_3 and u_4 are directed towards s_0. Let ρ be the initial configuration depicted here.



(b) Configuration obtained from ρ after a single cycle push on the cycle $\{(u_2, u_3); (u_3, u_2)\}$. This configuration and ρ are equivalent since one is obtained from the other by cycle pushing so they have the same Destination Forest. And as stated in Lemma 4.2.2 they have the same value $n_1 = 2$.



(c) Naturally, the Destination Forest (i.e. rotor configuration after maximal cycle push sequence) of ρ is equivalent to both previous configurations and the value n_1 remains the same.

Figure 4.2: Illustration of Lemma 4.2.2 and its proof. The red arcs in dashes represent the current rotor configuration.

Proof. A maximal cycle push sequence leads to an acyclic configuration, and we know that n_1 is preserved by cycle push. Since there is a unique forest with n_1 arcs directed towards s_1, namely the configuration where exactly $u_{n-(n_1-1)}, u_{n-(n_1-2)}, \ldots, u_n$ are directed towards s_1, this is the Destination Forest (see Figure 4.2c). □

The previous lemma enables us to compute in linear time the exit pattern of any configuration in a simple path graph – which easily solves **SP-ARRIVAL** simultaneously for all starting vertices.

4.2.2 . Routing Several Particles

Here we present how the previous results can be used to solve **MP-ARRIVAL** on a simple path graph. We only study the case of routing multiple particles in the case of the path graph, the general case remaining an open problem even for tree-like multigraphs.

It is well known that when we route any number of particles in a rotor graph up to the sinks, the final configuration does not depend on the order in which the particles move (which can alternate between particles) as long as we route the particles to the sinks (see [50]). This is also a consequence of Corollary 2.5.7.

Definition 4.2.3 (Equivalence Class for simple path graphs)**.** *For $0 \leq k \leq n$, let C_k be the rotor class of configurations ρ with $n_1(\rho) = k$ i.e., the set of configurations with exactly k vertices directed towards s_1.*

Note that the equivalence classes defined just above are exactly the equivalence classes of Lemma 4.1.5. We now explain the action of the movement of a particle on these classes. There are some results known in the general case for the action

of particle configurations on rotor configurations but we will present those in the last chapter of this document.

Theorem 4.2.4 (Group Action on simple path)**.** *Consider a configuration $\rho \in C_k$ with $0 \leq k \leq n$ and vertices u_i, $1 \leq i \leq n$. Processing a maximal rotor walk from (ρ, u_i) leads to a configuration $\rho' \in C_j$ where $j = (k + i) \mod n + 1$.*

Proof. By Theorem 4.1.6 we can as well suppose that ρ is acyclic, hence the k vertices directed towards s_1 are exactly $u_{n-k+1}, u_{n-k+2}, \ldots, u_n$.

- If $i \leq n - k$, then during the rotor walk, exactly i vertices change directions, hence $j = k + i$.
- If $i > n - k$, then $n - i + 1$ vertices change directions, hence $j = k - (n - i + 1) = k + i - (n + 1)$.

□

When combining the action of multiple particles on classes, we obtain the following theorem.

Theorem 4.2.5. *Let $\rho \in C_k$ be a rotor configuration. Consider the process where m particles are routed to the sinks from initial positions $i_1, i_2, \ldots, i_m$ in any (alternating or not) order. Then:*

- *The final configuration belongs to the class C_j with*

$$j = k + i_1 + i_2 + \cdots + i_m \mod n + 1,$$

- *exactly $p_1 = \left\lfloor \dfrac{k + (\sum_{t=1}^{m} i_t)}{n+1} \right\rfloor$ of these particles reach sink s_1, whereas $m - p_1$ reach s_0.*

Proof. By commutativity of routing operations, we can suppose that we fully route every particle to a sink before proceeding to the next one. Consider $\{0, 1, \ldots, n\}$ as a cycle of positions corresponding to the equivalence classes. We start from position k on the cycle, and every time a particle is routed from its initial position i_t to a sink, the final configuration belongs to C_j where j is obtained by moving from k on the cycle i_t times in cyclic ordering. It is contained in the proof of Theorem 4.2.4 that the particle ends in s_1 if and only if we move from position n to 0 during this process. Hence, we have to count the number of times this will happen, which is simply the integer quotient of the final position $k + i_1 + \cdots + i_m$ by the number of positions $n + 1$. □

This theorem enables us to solve **MP-ARRIVAL** on the path graph in linear time (even with an exponential number of particles, if arithmetic computations are made in constant time).

4.2.3 . The Return Flow with the Path Graph

The previous technique to compute the Destination Forest is based on an invariance property of cycle pushes, which is specific to the simple path graph. It does not generalize directly to other graphs, so we now give an equivalent formulation of the previous results. Instead of counting the number of vertices directed toward sinks, we now consider the number of vertices directed towards a given vertex u on each side of the path. To do this, we denote by $]u, u'[$ the set of vertices between u and u' in the order of vertices in the path, excluding u and u'.

Definition 4.2.6. *Let $\zeta_1(u)$ and $\zeta_0(u)$ be the number of vertices that are respectively directed towards u in $]u, s_1[$ and $]s_0, u[$.*

We now rewrite Theorem 4.2.2 with ζ_1 ans ζ_0 into Lemma 4.2.7.

Lemma 4.2.7. *For all non-sink vertices u_i, $\mathbf{S}_G(\rho, u_i)= s_1$ if and only if*

- *either $\zeta_1(u_i) < \zeta_0(u_i)$;*
- *or $\zeta_1(u_i) = \zeta_0(u_i)$ and u_i is directed towards s_1.*

Proof. We can decompose n_1 as follow:

$$n_1 = \underbrace{\zeta_0(u_i)}_{\text{vertices of }]s_0,u_i[} + \underbrace{I_1(u_i)}_{\text{Orientation of } u_i} + \underbrace{n - i - \zeta_1(u_i)}_{\text{vertices of }]u_i,s_1[}$$

where $I_1(u_i)$ is 1 if u_i is directed towards s_1 and 0 otherwise.

Now, from Theorem 4.2.2 we know that $\mathbf{S}_G(\rho, u_i) = s_1 \Leftrightarrow i \geq n - n_1 + 1$, which translates to

$$\zeta_0(u_i) + I_1(u_i) \geq \zeta_1(u_i) + 1,$$

hence the result. □

Remark. *Consider the following process: on the simple path graph, we remove the arc (u_i, u_{i-1}) so that the particle cannot go from u_i to $[s_0, u_i[$ anymore (in the case where $\rho(u_i) = (u_i, u_{i-1})$, we change it to $\rho(u_i) = (u_i, u_{i+1})$). Now, we put a particle on u_i and proceed to a maximal rotor walk. The number of times that the particle travels through arc (u_i, u_{i+1}) is exactly $\zeta_1(u_i) + 1$. This quantity is called the* return flow*, which will be defined in the general case in Theorem 4.3.6.*

4.3 . Tree-Like Multigraphs: Return Flow Definition

4.3.1 . Tree-Like Multigraphs

With a directed multigraph $G = (V, \mathcal{A}, h, t)$ we associate:

- A simple directed graph $\hat{G} = (V, \hat{\mathcal{A}})$ such that, for $u, v \in V$ there is an arc from u to v in $\hat{A}$ if there is at least one arc $a \in \mathcal{A}$ with $t(a) = u$ and $h(a) = v$. Please note that even if there are multiple arcs a that satisfy this property, there is only one arc with tail u and head v in $\hat{\mathcal{A}}$. As it is unique, an arc from u to v in $\hat{\mathcal{A}}$ will simply be denoted by (u, v).
- A simple undirected graph $\overline{G} = (V, E)$ such that, for $u, v \in V$ there is an edge between u and v in $\overline{G}$ if and only if there is at least one arc $a \in \mathcal{A}$ such that $t(a) = u$ and $h(a) = v$ or $h(a) = u$ and $t(a) = v$.

A vertex u for which $|\Gamma^+(u) \cup \Gamma^-(u)| = 1$ is called a *leaf*.

Definition 4.3.1 (Tree-Like Multigraph)**.** *We say that a multigraph G is* tree-like *if $\overline{G}$ is a tree.*

In this case, we define the leaves of G as the leaves of $\overline{G}$.

Definition 4.3.2 (Tree-Like Rotor Multigraph)**.** *A rotor multigraph $G = (V_0, S_0, \mathcal{A}, h, t, \theta)$ is tree-like if $(V, \mathcal{A}, h, t)$ is tree-like, and its set of leaves contains S_0.*

To avoid some complexity in the notation and proofs, we will only study stopping tree-like rotor multigraphs. We first show that the general case can be handled by reducing a non stopping instance to a stopping one.

We recall that a sink component is a strongly connected component in G such that there is no arc leaving the component. Recall that by [68], all sink components can be computed in linear time .

Lemma 4.3.3. *Consider a configuration ρ on a (not necessarily stopping) tree-like rotor multigraph $G = (V_0, S_0, \mathcal{A}, h, t, \theta)$. Consider a configuration ρ' on the* stopping *tree-like rotor multigraph $G' = (V_0', S_0', \mathcal{A}', h, t, \theta)$ where G' is obtained from G by replacing each sink component by a unique sink, and where $\rho'(u) = \rho(u)$ for each $u \in V_0'$. For any $u \in V$, finding the exit sink of u (if any) or the sink component reached by u in G for the configuration ρ can be directly determined by solving **SP-ARRIVAL** for the configuration ρ' in G'.*

Proof. Let ρ be a configuration in $\mathcal{R}(G)$ and $u \in V$ be a vertex.

- If the particle enters a sink component C while processing a rotor walk from (ρ, u) on G, then u has no exit sink for ρ. The same rotor walk in the graph G' ends in the sink that replaces C.
- If the particle reaches a sink s of G while processing a maximal rotor walk from (ρ, u) on G, then it does not enter a sink component of G hence the walk in G' is the same as in G.

□

Thanks to Lemma 4.3.3, we can work on graphs without sink components while keeping the generality of our results. Note that, after replacing sink components by sinks, the multigraph G' may no longer be a tree-like multigraph but a forest-like multigraph. However we can split the study of **SP-ARRIVAL** in each tree-like component of this forest since a particle cannot travel between those trees in a rotor walk.

4.3.2 . Return Flows

Let us consider the simple example depicted on 4.3 to motivate the introduction of (u, v)-subtrees and return flows, which is our main tool.

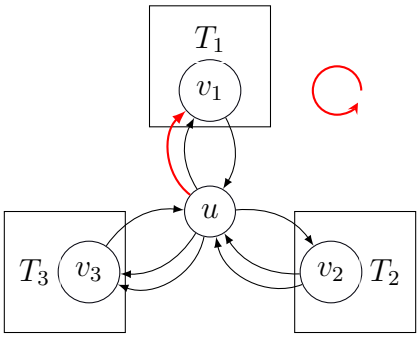


Figure 4.3: We sketch a stopping tree-like rotor multigraph as follows: a vertex u, its neighbours v_1, v_2, v_3 (that might be sinks), respectively belonging to T_1, T_2, T_3, the three connected components of $\bar{G} \setminus \{u\}$. In particular, we have $\hat{\mathcal{A}}^+(u) = \{(u, v_1); (u, v_2); (u, v_3)\}$. We consider the rotor configuration in red on u and θ_u is the anticlockwise order on the arcs of $\mathcal{A}^+(u)$.

In Figure 4.3, consider the routing of a particle starting at u:

1. the particle moves from u to v_1, and stays for a while in the subtree T_1 – where it either reaches a sink or comes back to u. Suppose it comes back to u. Then:

2. the particle moves from u to v_3, and either reaches a sink in T_3 or comes back to u. Suppose it comes back to u once again;

3. the rotor walk goes on, in T_3, then in $T_2, T_1, T_1, T_3, \ldots$

4. until finally the particle ends in a sink in one of the subtrees, say T_2.

Now consider only the relative movement that the particle had in T_2: it went from u into T_2 and back to u many times before it ended in a sink. If we were to replace T_1 and T_3 by a single arc leading back automatically to u, the relative movement in T_2 would have been exactly the same. The return flow will be a quantity that counts precisely the ability of each subtree to bounce back the particle to u. During

the process described above, every time the particle enters a subtree and returns to u, we can think of it as consuming a single unit of return flow in this subtree. The first time a particle enters a subtree with exactly one unit of return flow left, the particle must end in a sink of that subtree.

Definition 4.3.4 ((u,v)-subtree)**.** *Let $(u,v) \in \hat{\mathcal{A}}$. The* (u,v)-subtree $T_{(u,v)}$ *is a sub(multi)graph of G:*

- *whose vertices are all the vertices of the connected component of $\bar{G} \setminus \{u\}$ that contains v, together with u;*
- *whose arcs are all the arcs of G that link the vertices above, excepted in u where we remove all arcs of $\mathcal{A}^+(u)$ but a single arc a with head v. Such an arc a always exists because $(u,v) \in \hat{\mathcal{A}}$;*
- *whose rotor orders are unchanged except at u where $\theta_u(a) = a$.*

Such a subtree is a (not necessarily stopping) tree-like rotor multigraph. A rotor configuration ρ in G can be thought of as a rotor configuration ρ' in $T_{(u,v)}$ by defining that $\rho'(u) = a$ and $\rho'(w) = \rho(w)$ for all $w \in T_{(u,v)}$.

We define a notion of *flow* for a particular starting vertex.

Definition 4.3.5 (Flow of (u,v))**.** *We define the* flow *on arc $(u,v) \in \hat{\mathcal{A}}$ for configuration ρ, denoted by $F_\rho(u,v)$, the number of times (possibly infinite) that an arc with tail u and head v is visited during the maximal rotor walk of a particle starting from the rotor-particle configuration (ρ, u). We denote by $F_\rho(u)$ the flow vector of (u,v) for every $v \in \Gamma^+(u)$.*

Definition 4.3.6 (Return flow)**.** *The* return flow *of arc $(u,v) \in \hat{\mathcal{A}}$ for configuration ρ, denoted by $\zeta_\rho(u,v)$, is the flow on (u,v) in the (u,v)-subtree $T_{(u,v)}$.*

Note that the return flow $\zeta_\rho(u,v)$ also corresponds to the number of times the particle visits u while processing a maximal rotor walk from (ρ, u) in $T_{(u,v)}$ (see Figure 4.4). By definition of return flow, if $u \in S_0$, then $\zeta_\rho(u,v) = 0$, and if $v \in S_0$, or if $(v,u) \notin \hat{\mathcal{A}}$ then $\zeta_\rho(u,v) = 1$.

Remark also that, even if the tree-like multigraph is stopping, it is not necessarily the case of any (u,v)-subtree: this is for instance the case of a leaf v which is not a sink such that $(u,v) \in \hat{\mathcal{A}}$. Finiteness of the return flow characterizes the subtrees that are stopping as stated in Lemma 4.3.7.

Lemma 4.3.7. *Given a stopping tree-like multigraph G and $(u,v) \in \hat{\mathcal{A}}$, the (u,v)-subtree $T_{(u,v)}$ is stopping if and only if for any rotor configuration on G, the return flow of (u,v) is finite.*

Proof. If the (u,v)-subtree is stopping, then by Lemma 4.0.2 any rotor walk is finite, and the return flow is finite.

If the (u, v)-subtree is not stopping, there is a sink component C in $T_{(u,v)}$. If C does not contain u, it is also a sink component of G since we do not add or remove any arc in C while going from $T_{(u,v)}$ to G. But G is assumed stopping hence such sink component does not exist. The only possibility is that C contains u. In C, there is a vertex that will be visited infinitely often while processing a maximal rotor walk. But this will be the case for its neighbours in C as well, and, transitively, for every vertex in C. In particular u will be visited infinitely often, hence for any rotor configuration ρ, $\zeta_\rho(u, v)$ is infinite. □

We give a bound on the maximal value of the return flow in a multigraph as it will be used to express our complexity results later. This bound is a direct consequence of Lemma 3.1.5.

Lemma 4.3.8 (Return flow bound)**.** *Let $(u, u_1) \in \hat{\mathcal{A}}$ and ρ be a configuration. Then if there is a directed path $[u, u_1, \dots, u_k, s]$ from u to a sink $s \in S_0$ then $\zeta_\rho(u, u_1)$ satisfies*

$$\zeta_\rho(u, u_1) \leq e^{|\mathcal{A}|/e}$$

otherwise $\zeta_\rho(u, u_1)$ is infinite. In particular this shows that return flows can be written in at most $O(|\mathcal{A}|)$ bits.

Return flows and flows are linked by the following result:

Lemma 4.3.9. *Given a stopping tree-like rotor multigraph G, consider $u \in V_0$ and suppose that $h(D(\rho)(u)) = v$. Then:*

- $F_\rho(u, v) = \zeta_\rho(u, v)$ *;*
- *for all* $w \in \Gamma^+(u) \setminus \{v\}, \quad F_\rho(u, w) < \zeta_\rho(u, w)$*;*
- *for all* $w \in \Gamma^+(u) \cap \Gamma^-(u) \setminus \{v\}, \zeta_\rho(w, u) = F_\rho(u, w) + 1$*.*

Proof. Let ρ' be the configuration obtained after routing a particle from (ρ, u) until the particle is on u for the last time, in which case $h(\rho'(u)) = v$ and $\zeta_{\rho'}(u, v) = 1$. By definition of ρ', we have $F_{\rho'}(u, v) = 1$ and $F_{\rho'}(u, w) = 0$ for all $w \in \Gamma^+(u) \setminus \{v\}$. If, moreover, $w \in \Gamma^-(u)$ then $\zeta_{\rho'}(w, u) = 1 = 1 + F_{\rho'}(u, w)$. Hence $\zeta_{\rho'}(u, v) = 1 = F_{\rho'}(u, v)$ and by definition of the return flow, $\zeta_{\rho'}(u, w) \geq 1$ so that $\zeta_{\rho'}(u, w) > F_{\rho'}(u, w)$. Hence the property is satisfied for ρ'.

On the other hand it should be clear that $F_\rho(u, w) - F_{\rho'}(u, w) = \zeta_\rho(u, w) - \zeta_{\rho'}(u, w)$ for every $w \in \Gamma^+(u)$. If, moreover, $w \in \Gamma^-(u)$ and $w \neq v$ the previous quantity is also equal to $\zeta_\rho(w, u) - \zeta_{\rho'}(w, u)$. Hence the property is true for ρ as well. □

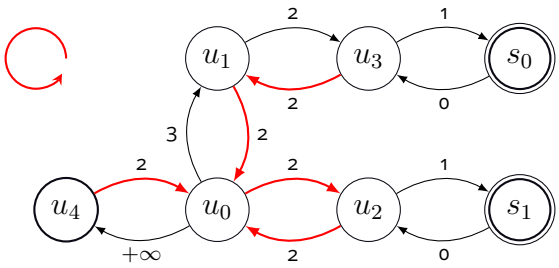


Figure 4.4: Examples of return flows in a simple graph. The rotor configuration is depicted by red arcs in dashes, with $S_0 = \{s_0, s_1\}$, and θ_{u_i} is the anticlockwise order on every vertex. We write the return flow of all arcs of $\hat{\mathcal{A}}$ next to their corresponding arc in $\mathcal{A}$. As a tutorial example, we detail the computation of $\zeta_\rho(u_1, u_0)$ and $\zeta_\rho(u_0, u_1)$. In the (u_1, u_0)-subtree, the particle will visit the following sequence of vertices $u_1, u_0, u_2, u_0, u_1, u_0, u_4, u_0, u_2, s_1$, where it crosses (u_1, u_0) twice, thus $\zeta_\rho(u_1, u_0) = 2$. For the (u_0, u_1)-subtree, the sequence of vertices visited by the particle is $u_0, u_1, u_0, u_1, u_3, u_1, u_0, u_1, u_3, s_0$ hence $\zeta_\rho(u_0, u_1) = 3$.

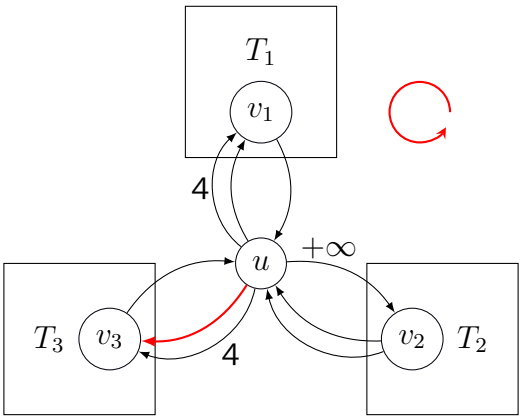


Figure 4.5: Consider the same stopping tree-like rotor multigraph as in Figure 4.3 where $\rho(u)$ is the red arc in dashes. The return flow of all arcs of $\hat{\mathcal{A}}^+(u)$ are given next to their corresponding arcs in $\mathcal{A}$.

4.3.3 . Revolving Routine

Based on the foregoing, using Lemma 4.3.9, in order to calculate $D(\rho)(v)$, we need to compute the flow of all arcs (u, w) with $w \in \Gamma^+(u)$ and compare it to $\zeta_\rho(u, w)$. This idea is introduced in the example drawn in Figure 4.5.

In Figure 4.5, let us put a particle on u and route it until it comes back to u. The first time, the particle will travel through the red arc in dashes and land on v_3. Then, the red arc in dashes (i.e., $\rho(u)$) is updated to $\theta_u(\rho(u))$ which is the other arc with tail u and head v_3. As $\zeta_\rho(u, v_3) > 1$, the particle does not reach a

sink in T_3, and the particle will come back to u after travelling in T_3. After this process (the particle's walk in T_3), the return flow of (u, v_3) has decreased by one as (u, v_3) has been crossed exactly once.
During the next step of this rotor walk, the particle will travel to v_3 again, make several moves in T_3 without reaching a sink and comes back to u. The return flow of (u, v_3) is now 2. Next, the particle will travel to v_2 and so on. At some point, the particle will travel through an arc with tail u and head v_i while the return flow of (u, v_i) is one. In this case, the particle will reach a sink in T_i and then will not come back to u. Here this condition is first met for $i = 3$ when the return flows of $(u, v_1), (u, v_2), (u, v_3)$ respectively are $2, +\infty, 1$ the last time the particle is on vertex u.

Inspired by this process, we state the following Theorem using notation $c(a, b)$ to denote the time complexity of dividing the number a by the number b.

Theorem 4.3.10. *For any vertex $u \in V_0$, given the return flows of all arcs $(u, v) \in \hat{A}$ with $v \in \Gamma^+(u)$, one can compute $D(\rho)(u)$ and the flow on each arc (u, v) in time*

$$O(|\Gamma^+(u)| \cdot c(\zeta_{\max}, |\mathcal{A}^+(u)|))$$

with ζ_{max} being the maximum (finite) value of $\zeta_\rho(u, v)$ for all $v \in \Gamma^+(u)$.

Proof. Algorithm 1 is a local routine that abstracts the process described in the caption of Figure 4.3. It computes $D(\rho)(u)$ as a and the flow of all arcs (u, v) as F. Notice that, in the routine and its improved version, we consider that $+\infty - 1$ and $+\infty/a$ with $0 < a < +\infty$ are equal to $+\infty$. Remark also that, since the input multigraph G of the routine is stopping, there is at least one return flow which is finite. However, since the return flow can be exponential as in the example drawn in Figure 3.2, Algorithm 1 might run in exponential time.

We can speed up the computation by noting that every time the rotor at u makes one full turn, we know exactly how many times each value of the flow F has increased. This remark leads to the improved Algorithm 2 denoted by IRR in the rest of the document. The IRR consists in two steps. First, we compute how many full turns the rotor on u does before the routine ends. Then, we use Algorithm 1.

At line 1 of IRR, computing $Q(v)$ is done in time $O\big(|\Gamma^+(u)| \cdot c(\zeta_{\max}, |\mathcal{A}^+(u)|)\big)$. Then, the loop at line 3 stops in at most $|\mathcal{A}^+(u)|$ steps, which is small compared to the first term. The same applies for the loop at line 6. In the end, IRR runs in time $O(|\Gamma^+(u)| \cdot c(\zeta_{\max}, |\mathcal{A}^+(u)|))$.

□

As the routines are crucial for the rest of the article, we now state an important monotony property. Here, we fix a vertex v in G and consider on the outgoing arcs of v the respective return flows ζ_1 and ζ_2 of two rotor configurations ρ_1 and ρ_2.

Algorithme 1 : Revolving Routine

input : u is a vertex of V; $\zeta : \Gamma^+(u) \to \mathbb{Z} \cup \{+\infty\}$ contains the return flows of arcs (u, v) of $\hat{\mathcal{A}}$ with $v \in \Gamma^+(u)$.

output : a an arc of $\mathcal{A}^+(u)$; F such that $F(v)$ is the number of times an arc of $\mathcal{A}^+(u)$ with head v has been visited

$F(v) \leftarrow 0$ for all $v \in \Gamma^+(u)$
$a \leftarrow \rho(u)$
while $\zeta(h(a)) > 1$ **do**
    $\zeta(h(a)) \leftarrow \zeta(h(a)) - 1$
    $F(h(a)) \leftarrow F(h(a)) + 1$
    $a \leftarrow \theta_u(a)$
return a, F

Algorithme 2 : Improved Revolving Routine (IRR)

input : u is a vertex of V; $\zeta : \Gamma^+(u) \to \mathbb{Z} \cup \{+\infty\}$ contains the return flows of arcs (u, v) of $\hat{\mathcal{A}}$ with $v \in \Gamma^+(u)$.

output : a an arc of $\mathcal{A}^+(u)$; F such that $F(v)$ is the number of times an arc of $\mathcal{A}^+(u)$ with head v has been visited

For $v \in \Gamma^+(u)$, let $Q(v)$ be the quotient of the euclidean division of $\zeta(v)$ by $|h^{-1}(v)|$; let $q_{\min}$ be the minimum value of $Q(v)$ for all v; let $R(v)$ be $\zeta(v) - (q_{\min} * |h^{-1}(v)|)$ and let $F(v) = 0$.
`/* This step corresponds to the return flow diminutions that occurs during the` $q_{\min} * |\mathcal{A}^+(u)|$ `first steps, and it also ensures that there exists at least one` v `such that in less than` $|\mathcal{A}^+(u)|$ `steps,` $R(v) < 1$. `*/`
$a \leftarrow \rho(u)$
while $R(h(a)) > 1$ **do**
    $R(h(a)) -= 1$
    $a \leftarrow \theta_u(a)$
for $v \in \Gamma^+(u)$ **do**
    $F(v) \leftarrow \zeta(v) - R(v)$
return *a,F*

Lemma 4.3.11 (Monotony of the flow)**.** *Let $u \in V_0$, and ρ_1, ρ_2 be two rotor configurations such that $\rho_1(u) = \rho_2(u)$. Let $v_i = h(D(\rho_i)(u))$ for $i \in \{1, 2\}$ and $\overline{\Gamma}$ be the set of vertices $v \in \Gamma^+(u)$ such that $\zeta_{\rho_1}(u, v) \geq \zeta_{\rho_2}(u, v)$. If $v_1 \in \overline{\Gamma}$, then*

- $F_{\rho_1}(u) \geq F_{\rho_2}(u)$ *component-wise,*
- $v_2 \in \overline{\Gamma}$.

Proof. Let ρ_i^k for $i \in \{1,2\}$ be the rotor configuration after the particle has visited u exactly k times during a maximal rotor walk starting from (ρ_i, u). In particular, $\rho_i^1 = \rho_i$. We denote by $R_i^k(u,v)$ and $F_i^k(u,v)$ the quantities $\zeta_{\rho_i^k}(u,v)$ and $F_{\rho_i^k}(u,v)$ respectively. Let K_i be the last time the particle is on u in this walk, which is characterized by $R_i^{K_i}(u,v_i) = 1$ and $h(\rho_i^{K_i}(u)) = v_i$ for $i \in \{1,2\}$.

For all $k \leq \min(K_1, K_2)$, let $\overline{\Gamma}^k$ be the set of vertices $v \in \Gamma^+(u)$ such that $R_1^k(u,v) \geq R_2^k(u,v)$. It turns out that in fact, for all such k we have $\overline{\Gamma}^k = \overline{\Gamma}$ since values of both R_1 and R_2 are decremented simultaneously.

We first show that $K_2 \leq K_1$. By contradiction, assume that $K_2 > K_1$. Since $v_1 \in \overline{\Gamma}$, then, at step K_1, $R_1^{K_1}(u,v_1) = 1 \geq R_2^{K_1}(u,v_1)$ and $h(\rho_i^{K_i}(u)) = v_1$ for $i \in \{1,2\}$. This implies that K_1 is the last time that the walk starting at (ρ_2, u) is at u, *i.e.,* $K_2 = K_1$, hence a contradiction.

For every $v \in \Gamma^+(u)$, we have $F_{\rho_i}(u,v) = \Delta F_i^k(u,v) + F_i^k(u,v)$ with $\Delta F_i^k(u,v) = F_{\rho_i}(u,v) - F_i^k(u,v)$. $\Delta F_i^k(u,v)$ is the number of times each arc (u,v) has been used until step k. Since $\rho_1(u) = \rho_2(u)$, and as long as $k \leq K_2$, $\Delta F_i^k(u,v)$ does not depend on i. It follows that $F_{\rho_1}(u,v) - F_{\rho_2}(u,v) = F_1^{K_2}(u,v) - F_2^{K_2}(u,v)$. If $v \neq v_2$ then $F_1^{K_2}(u,v) - F_2^{K_2}(u,v) = F_1^{K_2}(u,v) - 0 \geq 0$. Otherwise $F_1^{K_2}(u,v_2) - F_2^{K_2}(u,v_2) = F_1^{K_2}(u,v_2) - 1$. Since $K_1 \geq K_2$ and $h(\rho_1^{K_2}(u)) = v_2$, arc (u,v_2) will be used at least once more during the walk, *i.e.,* $F_1^{K_2}(u,v_2) \geq 1$. Hence the difference is positive which shows the first part of the lemma.

Let v be such that $v \notin \overline{\Gamma}$. Then $R_2^{K_2}(u,v) > R_1^{K_2}(u,v) \geq 1$. Hence $R_2^{K_2}(u,v) \geq 2$ which implies $v_2 \neq v$ and then $v_2 \in \overline{\Gamma}$. □

4.4 . SP-ARRIVAL for Tree-like Multigraphs

In this section we show that, for a given rotor configuration ρ on a multigraph G, we can compute the Destination Forest $D(\rho)$ in time complexity $O(|\mathcal{A}| \cdot c(\zeta_{\max}, |\mathcal{A}|))$, hence solve **SP-ARRIVAL** for every vertex at the same time. To achieve this, we recursively compute return flows for all arcs in $\hat{\mathcal{A}}$ and then use these flows to compute the destination forest.

In this section, let $G = (V_0, S_0, \mathcal{A}, h, t, \theta)$ be a stopping tree-like rotor multigraph and ρ be a rotor configuration on G.

The next two lemmas show how to compute the return flows by using Theorem 4.3.10.

Lemma 4.4.1. *For any two vertices u and v such that $(u,v) \in \hat{\mathcal{A}}$, and given $\zeta_\rho(v,w)$ for every $w \in \Gamma^+(v) \setminus \{u\}$, the return flow $\zeta_\rho(u,v)$ can be computed in time $O(|\Gamma^+(v)| \cdot c(\zeta_{\max}, |\mathcal{A}^+(v)|))$. We illustrate this operation in Figure 4.6.*

Proof. If $(v,u) \notin \hat{\mathcal{A}}$, then $\zeta_\rho(u,v) = 1$.

Otherwise, if $\zeta_\rho(v,w) = +\infty$ for all $w \in \Gamma^+(v)$ such that $w \neq u$, then $\zeta_\rho(u,v) = +\infty$.

In all other cases, apply the IRR to the vertex v, with input values $\zeta_\rho(v,w)$ for all $w \in \Gamma^+(v)$ such that $w \neq u$, and with $\zeta_\rho(v,u) = p$, where p is intended to be an integer large enough so that the output a of the IRR is such that $h(a) \neq u$. Then by Lemma 4.3.9, $\zeta_\rho(u,v) = F_\rho(v,u) + 1$, where $F_\rho(v)$ is obtained by the IRR. Parameter p in this proof should be chosen large enough so that variable $R(u)$ in the routine remains strictly positive; for instance p can be initialized with $(q_{\min} + 1) \cdot |h^{-1}(u)|$ with $q_{\min}$ defined in Algorithm 2 (IRR). □

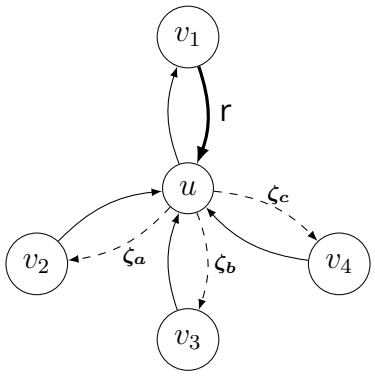


Figure 4.6: In this figure, we illustrate which value is computed with Theorem 4.4.1. If the return flows $\zeta_a, \zeta_b, \zeta_c$ are known, we can compute the return flow ζ.

.

Lemma 4.4.2. *For any vertex $u \in V_0$, and given $\zeta_\rho(u,v)$ for every $v \in \Gamma^+(u)$, one can compute the return flow of all arcs (w,u) with $w \in \Gamma^-(u)$ in time $O(|\Gamma^+(u)| \cdot c(\zeta_{\max}, |\mathcal{A}^+(u)|))$. We illustrate this operation in Figure 4.7.*

Proof. For all $w \in \Gamma^-(u) \setminus \Gamma^+(u)$, we have $\zeta_\rho(w,u) = 1$.

We use Theorem 4.3.10 once on u to compute $D(\rho)(u)$ and the vector flow $F_{\rho(u)}$. Let $v = h(D(\rho)(u))$. Then, by Lemma 4.3.9, for all $w \in \Gamma^-(u) \cap \Gamma^+(u) \setminus v$, we have $\zeta_\rho(w,u) = F_\rho(u,w) + 1$. It remains to apply Lemma 4.4.1 once more to compute $\zeta_\rho(v,u)$. All in all, we have used Lemma 4.3.10 twice, hence the complexity. □

One can check by applying Lemma 4.4.2 that, on the example of Figure 4.5, we have $\zeta_\rho(v_1,u) = 3$, $\zeta_\rho(v_2,u) = 2$ and $\zeta_\rho(v_3,u) = 4$.

We are now ready to state our main theorem. Complexity bounds are given in two different contexts:

- a context where the time needed for arithmetic computation matters, as in a Turing machine, using notation $c(a,b)$;

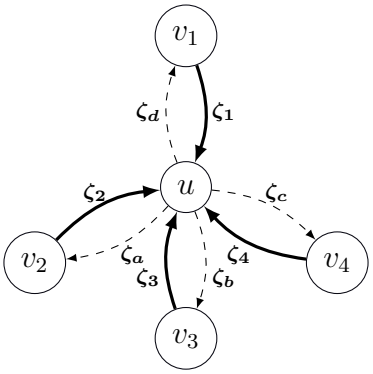


Figure 4.7: In this figure, we illustrate which values is computed with Theorem 4.4.2. If the return flows $\zeta_a, \zeta_b, \zeta_c, \zeta_d$ are known, we can compute the return flows $\zeta_1, \zeta_2, \zeta_3, \zeta_4$.

.

- another context where arithmetic operations can be done in constant time, where we achieve linear complexity in the size of the graph.

Theorem 4.4.3 (Complete Destination Algorithm)**.** *The configuration $D(\rho)$ can be computed in time $O(|\mathcal{A}|)$ for a stopping tree-like multigraph in a model where arithmetic operations can be made in constant time, or alternatively in $O(|\mathcal{A}| \cdot c(\zeta_{\max}, |\mathcal{A}|))$ on a bounded RAM machine.*

Proof. Consider an arbitrary vertex x. We proceed to a Breadth-First Search (BFS) starting from x in $\overline{G}$, the simple undirected graph associated with G. Let $e_1, e_2, \ldots, e_m$ (resp., $u_0, u_1, \ldots, u_k$ with $u_0 = x$) be the prefix order on the edges (resp., on the vertices) of $\overline{G}$ obtained during the BFS, i.e., the order in which the edges (resp., the vertices) are visited.

The algorithm is split into two phases:

1. Computation of return flows for all arcs directed from x towards the leaves: for $t = m, m-1, \ldots, 2, 1$, if the edge e_t corresponds to an existing arc (u_i, u_j) of $\hat{\mathcal{A}}$, such that (u_i, u_j) is directed from x towards a leaf of G, consider two cases. Firstly, if u_j is a leaf, then store that either $\zeta_\rho(u_i, u_j) = 1$ or $\zeta_\rho(u_i, u_j) = +\infty$ depending on whether $u_j \in S_0$ or not. Secondly, if u_j is not a leaf, then by definition of a BFS and a prefix order, if an arc (u_j, v) with $v \neq u_i$ corresponds to an edge $e_{t'}$, then $t < t'$. Hence, we already know the value of $\zeta_\rho(u_j, v)$ for every $v \neq u_i$ with $v \in \Gamma^+(u_j)$. This means that we can compute recursively $\zeta_\rho(u_i, u_j)$ by Lemma 4.4.1.

2. Computation of return flows of all arcs directed from the leaves towards x: when this phase begins, for any vertex u except x, all return flows $\zeta_\rho(u, v)$ for all $v \in \Gamma^+(v)$ are known, excepted the return flow of the

arc directed from u to x. We use Lemma 4.4.2 applied to vertices u_i in increasing order on i as it guarantees that the conditions to apply Lemma 4.4.2 are met. Furthermore, we also compute $D(\rho)(u_i)$ at the same time.

The time needed for the BFS part is $O(|\hat{\mathcal{A}}| + |V|)$.

In the first phase, we use Theorem 4.3.10 at most once for every vertex v, for the arc (u, v) coming from x towards the leaves. During the second phase, we we use Theorem 4.3.10 at most twice for each vertex. All in all, we use Theorem 4.3.10 three times for each vertex. Hence the time complexity is $O(\sum_{v \in V} (|\hat{\mathcal{A}}^+(v)| \cdot c(\zeta_{\max}, |\mathcal{A}|)))$ which amounts to $O(|\mathcal{A}| \cdot c(\zeta_{\max}, |\mathcal{A}|))$. □

Remark. *From [**cook1972**], the execution time on a Turing machine simulating the bounded RAM machine is bounded by* $O(|\mathcal{A}| \cdot c(\zeta_{\max}, |\mathcal{A}|))^3$.

We showed in Theorem 4.3.8 that return flows could be written in at most $O(|\mathcal{A}|)$ bits which gives an upper bound for $c(r_{\max}, |\mathcal{A}|)$ of $k|\mathcal{A}| \log(|\mathcal{A}|)$ for some constant $k > 0$. It is proved in [48] that the multiplication of two n bits integers can be done in time $O(n \log(n))$ and as the complexity of the division is equivalent to the complexity of multiplication (see [17]), the bound follows. Thus the complexity of our algorithm is $O(|\mathcal{A}|^2 \log(|\mathcal{A}|))$ in this context.

4.5 . One-player Rotor Game

Problem **SP-ARRIVAL** can be seen as a zero-player game where the winning condition is that the particle reaches a particular sink (or set of sinks). The one and two player variants of **SP-ARRIVAL** (i.e., deterministic analogs of *Markov decision processes* and *Stochastic games*) we address in the next sections are inspired from [77], but differ by the choice of the set of strategies (see the discussion hereafter).

In this section, we specifically consider a game with a single player that controls a subset of vertices $V_{\mathcal{MAX}}$ of V_0. Given a rotor configuration on the rest of the vertices of V_0, a starting vertex and an integer value for each sink, his goal is to wisely choose the initial rotor configuration of the vertices he controls (his strategy) such that the particle reaches one of the sinks with maximal value.

A remark is in order here: in the seminal paper [77], a strategy is defined in a more general way since it consists in choosing an outgoing-arc each time the particle is on a vertex controlled by the player. In particular, for a given vertex, the sequence of arcs may not follow a rotor rule, and the number of strategies is even unbounded. It has been shown in that paper that solving such game is NP-complete. On the one hand, the given reduction of 3-SAT can easily be adapted to fit to our framework showing that our definition of the game, although simpler since the set of strategies is finite, still leads to an NP-complete problem. On the other hand, our results extend naturally to general strategies, but at the cost of

more technicalities. For instance, the use of general strategies may lead to non-stopping rotor graphs even if every vertex is connected to a sink. This case also seems to us a very natural extension of the zero player case.

To formally define the game, we introduce the following definition.

Definition 4.5.1 (Partial Configuration)**.** *Let V' be a subset of V_0, a* partial rotor configuration *on V' is a mapping ρ' from V' to $\mathcal{A}$ such that $\rho'(u) \in \mathcal{A}^+(u)$ for all $u \in V'$.*

A one-player rotor game (resp., one-player tree-like rotor game) is given by $(V_r, V_{\mathcal{MAX}}, S_0, \mathcal{A}, h, t, \theta, \text{val}, \rho)$ where V_r, $V_{\mathcal{MAX}}$ and S_0 are disjoint sets of vertices, such that

- $(V_0, S_0, \mathcal{A}, h, t, \theta)$ is a rotor graph (resp., tree-like rotor graph) with $V_0 = V_r \cup V_{\mathcal{MAX}}$;
- val is a map from S_0 to $\mathbb{N}$ corresponding to a utility of the player who wants the particle to reach a sink s with the highest possible value $\text{val}(s)$;
- ρ is a partial configuration on V_r, the initial configuration on the vertices not controlled by the player.

The tree-like rotor game is *stopping* if and only if the induced rotor graph $(V_0, S_0, \mathcal{A}, h, t, \theta_v)$ is stopping.

The player is called $\mathcal{MAX}$, and a *strategy* for $\mathcal{MAX}$ is a partial rotor configuration on $V_{\mathcal{MAX}}$. We denote by $\Phi_{\mathcal{MAX}}$ the finite set of strategies for this player.

Consider a partial rotor configuration ρ on V_r together with strategy ϕ and denote by (ρ, ϕ) the rotor configuration where we apply the partial configuration ρ or ϕ depending on whether the vertex is in V_r or $V_{\mathcal{MAX}}$.

The *value of the game* for strategy ϕ and starting vertex u_0 is denoted by $\text{val}_\phi(u_0)$ and is equal to $\text{val}(s)$ where s is the sink reached by a maximal rotor walk from the rotor particle configuration $((\rho, \phi), u_0)$ if any, and 0 otherwise. As in the zero-player framework, up to computing strongly connected components that do not contain sinks and replacing each of them with a sink of value 0, we can suppose that the tree-like rotor game is stopping. In the following, all rotor games we consider are tree-like and stopping unless stated otherwise.

When u_0 is fixed, the maximal value of $\text{val}_\phi(u_0)$ over all strategies $\phi \in \Phi_{\mathcal{MAX}}$ is called the *optimal value* of the game with starting vertex u_0 and is denoted by $\text{val}^*(u_0)$. Any strategy $\phi \in \Phi_{\mathcal{MAX}}$ such that $\text{val}_\phi(u_0) = \text{val}^*(u_0)$ is called an *optimal strategy* for the game starting in u_0. Observe that optimal strategies may depend on the choice of u_0 as illustrated in Figure 4.8. The one-player **SP-ARRIVAL** problem consists in computing the optimal value of a given starting vertex in a one-player rotor game.

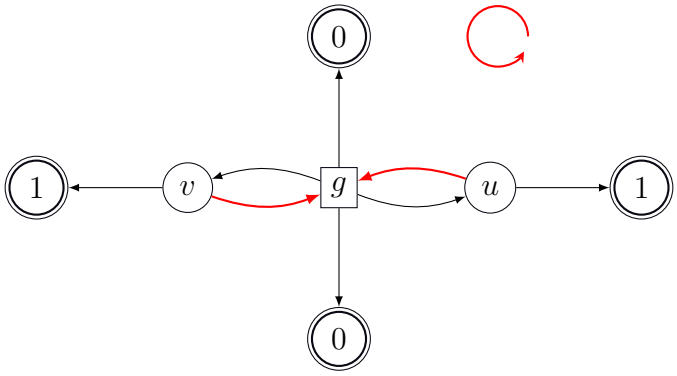


Figure 4.8: Simple graph where the optimal strategy depends on the starting vertex u_0, with $V_{\mathcal{MAX}} = \{g\}$, with $u, v \in V_r$ and with all other vertices being sinks. As in previous examples, the starting configuration is depicted by red arcs in dashes, and the rotor order on all vertices is an anticlockwise order on their outgoing arcs. In the case $u_0 = v$, the only optimal strategy is $\sigma(g) = (g, v)$ and the game has value 1. The case $u_0 = u$ and $u_0 = v$ are symmetrical.

First, we study the case where the values of the sinks are binary, then we adapt those results to the case of nonnegative integer values, and finally, we present our results for some different set of strategies.

4.5.1 . One-player Binary Rotor Game

In this subsection, we restrict the game to the case where values of sinks are binary numbers i.e., $\mathrm{val}(s) \in \{0, 1\}$ for all $s \in S_0$. Recall that in the tree-like rotor graph, $T_{(u,v)}$ denotes the (u, v)-subtree. We extend this notation to denote the one-player, not necessarily stopping, game played on the (u, v)-subtree where we restrict $V_{\mathcal{MAX}}$ and V_r to the subtree. For this game, we only consider the case where the starting vertex is u.

Definition 4.5.2 (Value under strategy)**.** *Let (u, v) be an arc of $\hat{\mathcal{A}}$. Given a strategy ϕ for the (u, v)-subtree, we denote by $\mathrm{val}_\phi(u, v)$ (resp., $\mathrm{val}^*(u, v)$) the* value *of the game* under strategy ϕ *(resp., under an optimal strategy) in $T_{(u,v)}$. This is called the value (resp., the* optimal value*) of the arc (u, v) for strategy ϕ.*

Definition 4.5.3 (Optimal return flow ζ^*)**.** *Let (u, v) be an arc of $\hat{\mathcal{A}}$. If $\mathrm{val}^*(u, v) = 0$, then $\zeta^*(u, v)$ is defined as the maximum of $\zeta_\phi(u, v)$ over all strategies ϕ on $T_{(u,v)}$, otherwise it is the minimum of $\zeta_{\phi^*}(u, v)$ among* optimal *strategies ϕ^* on $T_{(u,v)}$.*

The next lemma connects the value $\mathrm{val}_\phi(u)$ with the value of the last outgoing arc of vertex u while processing a maximal rotor walk from the rotor-particle configuration $((\rho, \phi), u)$.

Lemma 4.5.4. *Let a be an arc of $\mathcal{A}^+(u)$ such that $D(\rho,\phi)(u) = a$ with $h(a) = v$. We have $\mathrm{val}_\phi(u) = \mathrm{val}_\phi(u,v)$.*

To recursively compute an optimal strategy, we need a stronger notion of optimality, namely a *subtree optimal strategy*.

Definition 4.5.5 (Subtree optimal strategy)**.** *A strategy ϕ^* is* subtree optimal *at u_0 if it is optimal at u_0 and, moreover, $\mathrm{val}_{\phi^*}(u,v) = \mathrm{val}^*(u,v)$ and $\zeta_{\phi^*}(u,v) = \zeta^*(u,v)$ for every (u,v)-subtree such that (u,v) is directed from u_0 towards the leaves.*

Instead of recursively computing only the return flow as in the zero-player game, we now propagate both the optimal value and the optimal return flow to construct a subtree optimal strategy. Here, we give an equivalent to Lemma 4.4.1 for the one-player game that details how to recursively compute val^* and ζ^*.

Lemma 4.5.6. *Let $(u,v) \in \hat{\mathcal{A}}$. For every (v,w)-subtree with $w \in \Gamma^+(v) \setminus \{u\}$, assume that there is a strategy ϕ^*_w that is subtree optimal. Let ϕ_v be a strategy on the (u,v)-subtree $T_{(u,v)}$ such that $\phi_v(z) = \phi^*_w(z)$ when $z \in T_{(v,w)} \cap V_{\mathcal{MAX}}$ and $z \neq v$. Furthermore, if $v \in V_{\mathcal{MAX}}$, let Φ_v be the set of strategies defined on $T_{(u,v)}$ that agree with ϕ_v on every vertex but v. We consider two cases:*

- *if $\mathrm{val}^*_{(u,v)} = 0$, $\phi_v(v)$ is a strategy in Φ_v that maximizes the return flow on (u,v);*
- *if $\mathrm{val}^*_{(u,v)} = 1$, $\phi_v(v)$ is a strategy in Φ_v that is optimal and minimizes the return flow on (u,v).*

Then ϕ_v is subtree optimal on $T_{(u,v)}$.

Proof. We suppose that $v \in V_{\mathcal{MAX}}$. The case $v \notin V_{\mathcal{MAX}}$ can be treated similarly and is omitted.

By assumption, the restriction of ϕ_v to every subtree $T_{(w,z)}$, where (w,z) is an arc of $T_{(u,v)}$ different from (u,v), and directed from u towards the leaves, is subtree optimal. It remains to show that $\mathrm{val}_{\phi_v}(u,v) = \mathrm{val}^*(u,v)$ and $\zeta_{\phi_v}(u,v) = \zeta^*(u,v)$.

- Assume that $\mathrm{val}^*(u,v) = 0$. In this case, we have $\mathrm{val}_{\phi_v}(u,v) = \mathrm{val}^*(u,v) = 0$ as for any strategy.

  For the return flow, we consider different cases.

  If $(v,u) \notin \hat{\mathcal{A}}$, then $\zeta_\phi(u,v) = 1$ for every strategy ϕ on $T_{(u,v)}$, so $\zeta_{\phi_v}(u,v) = 1 = \zeta^*(u,v)$ is maximal. If $(v,u) \in \hat{\mathcal{A}}$ and $\zeta^*(v,w) = +\infty$ for every $w \in \Gamma^+(v) \setminus \{u\}$, then $\zeta_\phi(u,v) = +\infty$ for every strategy ϕ on $T_{(u,v)}$, in particular $\zeta_{\phi_v}(u,v) = +\infty = \zeta^*(u,v)$.

  Finally, if $(v,u) \in \hat{\mathcal{A}}$ and there is w such that $\zeta^*(v,w) < +\infty$, let $w_0 = h(D(\rho,\phi_v)(v))$ (defined in the stopping rotor graph $T_{(u,v)}$) and consider

a strategy ϕ defined on $T_{(u,v)}$ with $\phi(v) = \phi_v(v)$. By Lemma 4.5.4, we have $\mathsf{val}_{\phi_v}(v) = \mathsf{val}_{\phi_v}(v, w_0)$ hence $\mathsf{val}_{\phi_v}(v, w_0) = 0$. Since ϕ_v is subtree optimal on the (v, w_0)-subtree, it follows that $\mathsf{val}^*(v, w_0) = 0$ and then $\mathsf{val}_\phi(v, w_0) = 0$ and finally we have $\zeta_\phi(v, w_0) \leq \zeta_{\phi_v}(v, w_0)$.

We have $\phi(v) = \phi_v(v)$ and w_0 is such that $\zeta_\phi(v, w_0) \leq \zeta_{\phi_v}(v, w_0)$. Following Lemma 4.3.11 applied to v, $w_0 \in \overline{\Gamma}$, and then $F_\phi(v, u) \leq F_{\phi_v}(v, u)$. By Lemma 4.3.9, we have $\zeta_\phi(u, v) = F_\phi(v, u)+1$ and $\zeta_{\phi_v}(u, v) = F_{\phi_v}(v, u)+1$. This implies that $\zeta_\phi(u, v) \leq \zeta_{\phi_v}(u, v)$. Since $\phi_v(v)$ is chosen so that to maximize the return flow on the set of strategies Φ_v the result follows.

- Assume that $\mathsf{val}^*(u, v) = 1$. We first show that there is an optimal strategy in Φ_v.

  For this, consider an optimal strategy ϕ^* on $T_{(u,v)}$ and let $w_0 = h(D(\rho, \phi^*)(v))$. By Lemma 4.5.4, it follows that $\mathsf{val}_{\phi^*}(v, w_0) = 1$ and then $\mathsf{val}^*(v, w_0) = 1$. Let ϕ be the strategy in Φ_v such that $\phi(v) = \phi^*(v)$. Since ϕ is subtree optimal on $T_{(v,w_0)}$ we have $\zeta_{\phi^*}(v, w_0) \geq \zeta_\phi(v, w_0)$. On the other hand, let W be the set of vertices $w \in \Gamma^+(v) \setminus \{u\}$ such that $\mathsf{val}^*(v, w) = 0$. On this set $\zeta_{\phi^*}(v, w) \leq \zeta_\phi(v, w)$, *i.e.,* $w \notin \overline{\Gamma}$ following the notation of Lemma 4.3.11. Hence $D(\rho, \phi)(v) \notin W$ which implies $\mathsf{val}_\phi(u, v) = 1$.

  Now, showing that $\zeta_{\phi_v}(u, v) = \zeta^*(u, v)$ is done exactly the same way as the case $\mathsf{val}^*(u, v) = 0$.

□

Note that if $v \in S_0$, then there is no decision to make in $T_{(u,v)}$, and the empty strategy is subtree optimal. Otherwise, Lemma 4.5.6 shows inductively that such subtree optimal strategy exists for any $T_{(u,v)}$ where $(u, v) \in \hat{\mathcal{A}}$.

As a second remark, Lemma 4.5.6 can straightforwardly be adapted to the case where the player seeks to minimize the value, by swapping the role of subtrees of value 0 and 1. This will be used in next section when we consider a two-player game.

However, this process requires to determine $\phi_v(v)$ which minimizes or maximizes (depending of the optimal value of arc (u, v)) the return flow. To avoid an additional $|A^+(v)|$ factor in the time complexity by trying all possible choices for $\phi_v(v)$, we propose Algorithm 3 which runs in time $O(|\Gamma^+(v)| \cdot c(\zeta_{\max}, |\mathcal{A}|))$. In this algorithm, a_s is an arc that will try all possible starting configurations on v in the IRR ; whereas a_e is the corresponding output arc, i.e., the destination arc if we start in a_s. The important fact here is that, when a_s in incremented by θ_v, a_e possibly moves in the cyclic ordering but can never make a full turn and go beyond a_s; this is because the loop part of the IRR never makes a full turn. During this process, we just keep track of the maximum and minimum return flows depending on the value of the subtree in the direction a_e.

Now, in the same spirit as for the zero-player game, we can use Lemma 4.5.6 as the basis of a recursive algorithm for computing val$^*(u,v)$ and $\zeta^*(u,v)$ for all $(u,v) \in \hat{\mathcal{A}}$ directed from v_0 towards the leaves. This leads to our main theorem.

Theorem 4.5.7 (Computation of val$^*(u_0)$)**.** *The optimal value val$^*(u_0)$ can be computed in the same time complexity as the computation of $D(\rho)$ in the zero-player game (see Theorem 4.4.3).*

Proof. By using Lemma 4.5.6, we recursively compute val$^*(u,v)$ and $\zeta^*(u,v)$ for all arcs $(u,v) \in \hat{\mathcal{A}}$ such that (u,v) is directed from u_0 towards the leaves. Several cases are considered:

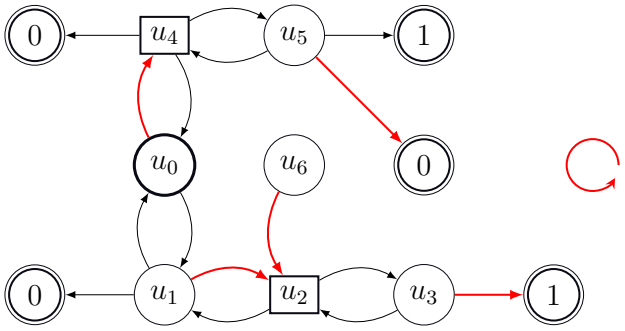
- if $v \in S_0$ then val$^*(u,v) =$ val(v) and $\zeta^*(u,v) = 1$;
- otherwise we run Algorithm 3.

This process is done at most once for each vertex v which results in the time complexity given in the statement of the theorem.

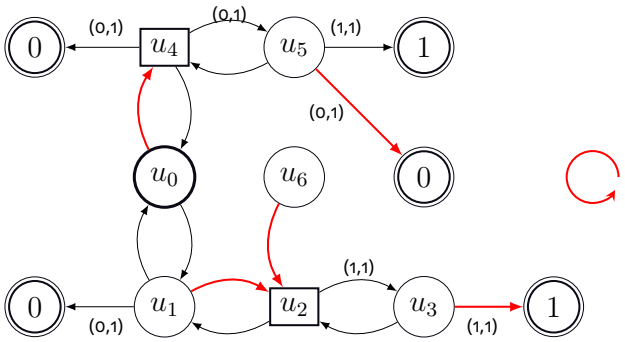
It remains to compute val$^*(u_0)$ knowing val$^*(u_0,w)$ and $\zeta^*(v_0,w)$ for every $w \in \Gamma^+(u_0)$. For this, we run Algorithm 3 on arc (z,u_0) with z being a fictive vertex such that the only arc incident to z is (z,u_0), r and val are $\zeta^*(u_0,w)$ and val$^*(u_0,w)$ respectively for every $w \in \Gamma^+(u_0)$. Then the binary value returned by the algorithm is val$^*(z,u_0)$. But in the (z,u_0)-subtree, the first step leads the particle to vertex u_0 and then never goes back to z: the run is then similar to the run starting at u_0 in the tree-like rotor game. Hence val$^*(z,u_0) =$ val$^*(u_0)$.

□

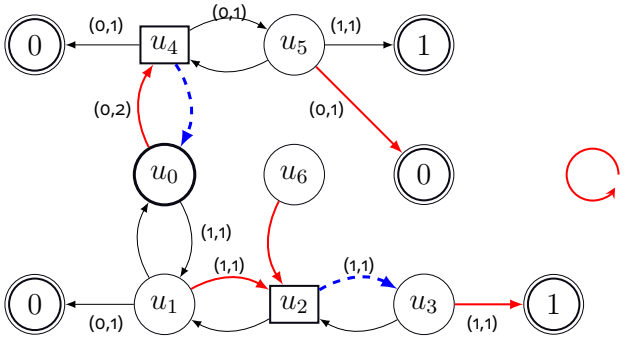
Some remarks are in order here. (i) Optimal Strategy Computation Algorithm used to compute val$^*(u_0)$ provides the optimal value of u_0 as well as a subtree optimal strategy at u_0 for every decisional vertex. A tutorial example is given in Figure 4.9. (ii) In the case of a simple graph, one can compute val$^*(u_0)$ for every vertex u_0 with the same time complexity as in Theorem 4.5.7 as detailed in Section 4.7.



(a) Consider the one-player tree-like rotor game above with u_0 being the starting vertex, $V_{\mathcal{MAX}} = \{u_2, u_4\}$ (depicted by squares), $V_r = \{u_0, u_1, u_3, u_5, u_6\}$, ρ being the partial configuration on vertices of V_r (depicted by the red arcs in dashes) and θ being the anticlockwise order on the outgoing arcs of each vertex (depicted by the cycling arrow on the right). The set S_0 contains all other vertices. Each of these sinks is represented by two circles with its value written inside.



(b) We run the first 8 steps that does not involve a positional vertex of our algorithm recursively from the leaves and write the couple $(\text{val}^*(u,v), \zeta^*(u,v))$ next to each arc (u,v) directed from u_0 towards the leaves.



(c) We proceed three more steps of our algorithm and compute $\sigma^*(u_2)$ and $\sigma^*(u_6)$ while doing so. The strategy σ^* is depicted by blue arcs. From here, we deduce that $\text{val}^*(u_0) = 1$ thanks to Theorem 4.5.7.

Figure 4.9: Computation of val^* and ζ^* using Theorem 4.5.7.

4.5.2 . One-player Integer Rotor Game

We now turn to the case where val(s) is no longer restricted to be a binary value. The main difference with the binary case is that there may not exist a subtree optimal strategy as illustrated in Figure 4.10. In this example the value 1 is an intermediate sink value (neither maximal not minimal), hence it cannot be decided with the only knowledge of the subtree optimal strategy on the (u_0, u)-subtree whether the return flow should be minimized in order to try reaching this sink or maximized if a sink with higher value can be reached in the rest of the graph. The knowing of val$^*(u, v)$ and $\zeta^*(u, v)$ that was enough in the binary case, is not sufficient anymore for computing the optimal value recursively. In the case of simple graphs, we show in Section 4.7.2.2 that we can add information on the subtrees in order to compute the optimal value in linear time complexity, but this technique does not extend to multigraphs.

Indeed, in the binary case we had two types of sinks those with value 0 and with value 1 which are respectively indistinguishable. Hence, reaching a sink of value 1 ensures that there is no sink of greater value in the graph. But, for the integer case, reaching a sink with value x does not guarantee that there is no sink with greater value x' elsewhere in the graph. Therefore, the algorithm that worked for binary values does not apply here.

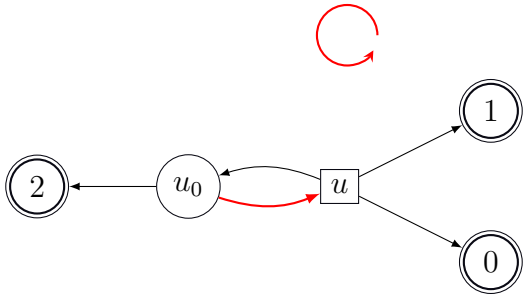


Figure 4.10: Example of a tree-like rotor game that does not admit a subtree optimal strategy. Here, $u \in V_{\mathcal{MAX}}$ and $u_0 \in V_r$. All other vertices belong to S_0 and their value is written inside them. The initial configuration of u_0 is the red arc. In the (u_0, u)-subtree, the only optimal strategy is to direct u towards the sink of value 1 which gives value 1 with return flow 1. In the entire game with starting vertex u_0, the value of this strategy is 1. But the optimal value is 2 which is obtained by directing u towards the sink of value 2.

Despite that, Theorem 4.5.7 can be used as a basis of a binary search (dichotomy) method for computing the optimal value. For this, consider the decision problem of determining whether a sink of value at least x can be reached. We can solve it by introducing a binary game obtained by replacing all values that are greater or equal to x by one and the others by zero. It should be clear that the value of the binary game is one if and only if a sink of value at least x can

be reached in the initial game. All in all, the non-binary game can be solved in $O(\log(|S_0|))$ such iterations. Knowing the optimal value, say v^*, an optimal strategy can be computed by solving the binary associated game where threshold x is chosen equal to v^*.

4.5.3 . One-player Rotor Game: Other Set of Strategies

Here we briefly discuss some variants of this game, where the player can choose from a different set of strategies than just the initial configurations on the vertices of $V_{\mathcal{MAX}}$.

1. Let us consider the set of strategies where the player can freely decide the rotor order on vertices he controls and the starting configuration on it. Consider an arc $(u, v) \in \hat{\mathcal{A}}$ and assume that $\zeta^*(v, w)$ and $val^*(v, w)$ are known for every $w \in \Gamma^+(v) \backslash u$. One can compute a subtree optimal strategy on the (u, v)-subtree in the same way as for the previous binary case but where the return flows are either $\zeta_\phi(u, v) = q_{min} * |\mathcal{A}_{(u,v)}|$ if $\mathsf{val}^*(u, v) = 1$ (all occurrences of an arc with head u are placed at the end of the rotor order) or $\zeta_\phi(u, v) = (q_{min} + 1) * |\mathcal{A}_{(u,v)}|$ if $\mathsf{val}^*(u, v) = 0$ (all occurrences of an arc with head u are placed at the beginning of the rotor order). This also simplifies the integer case consequently.

2. Let us consider the infinite set of strategies where the player can choose at each step of the rotor walk the orientation of the vertices he controls (as in [77]). For a vertex $v \in V_{\mathcal{MAX}}$, the player can choose to put the return flow on any outgoing arc of v to either 1 or $+\infty$. Consider an arc $(u, v) \in \hat{\mathcal{A}}$ and assume that $\zeta^*(v, w)$ and $val^*(v, w)$ are known for every $w \in \Gamma^+(v) \setminus u$. A subtree optimal strategy is easily computed by choosing the strategy to always go towards a vertex w if $\mathsf{val}^*(v, w) = 1$ and if not to choose the strategy that maximizes the value of $\zeta_\phi(u, v)$. In particular, if there exists an arc $(v, u) \in \hat{\mathcal{A}}$ the strategy on v would be to always go towards (v, u). Once again, the integer case is simplified consequently.

4.6 . Two-player Rotor Game

We now consider a two-player, zero-sum version of the zero-player game, where players control distinct subsets of vertices of V_0, one trying to maximize the value of the sink that has been reached whereas the other one tries to minimize it. A similar game (but where players freely decide the orientation of their vertices at each time step) has been studied in [77] and shown to be P-SPACE hard.

More formally, a two-player rotor game is given by

$$G = (V_r, V_{\mathcal{MAX}}, V_{\mathcal{MIN}}, S_0, \mathcal{A}, h, t, \theta, \mathsf{val}, \rho)$$

where $V_r, V_{\mathcal{MAX}}, V_{\mathcal{MIN}}$ and S_0 are disjoint sets of vertices, such that:

- for all partial configurations τ on $V_{\mathcal{MIN}}$,

$$G(\cdot, \tau) = (V_r \cup V_{\mathcal{MIN}}, V_{\mathcal{MAX}}, S_0, \mathcal{A}, h, t, \theta, \mathsf{val}, (\rho, \tau))$$

  is a one-player rotor game;

- for all partial configurations ϕ on $V_{\mathcal{MAX}}$,

$$G(\phi, \cdot) = (V_r \cup V_{\mathcal{MAX}}, V_{\mathcal{MIN}}, S_0, \mathcal{A}, h, t, \theta, \mathsf{val}, (\rho, \phi))$$

  is a one-player rotor game.

If all the one-player games are tree-like (in other words, if the underlying graph is tree-like) then the two-player game is also said to be tree-like.

A strategy for player $\mathcal{MAX}$ (respectively player $\mathcal{MIN}$) is a partial rotor configuration on $V_{\mathcal{MAX}}$ (respectively on $V_{\mathcal{MIN}}$). We denote by $\Phi_{\mathcal{MAX}}$ and $\Phi_{\mathcal{MIN}}$ the sets of strategies for these players. When τ is fixed, $\mathcal{MAX}$ tries to maximize the value of the final sink in $G(\cdot, \tau)$; whereas when ϕ is fixed, $\mathcal{MIN}$ tries to minimize it in $G(\phi, \cdot)$.

The *value of the game* for strategies ϕ, τ and starting vertex u_0 is denoted by $\mathsf{val}_{\phi,\tau}(u_0)$ and is the value $\mathsf{val}(s)$ where s is the sink reached by a maximal rotor walk from the rotor particle configuration $((\rho, \phi, \tau), u_0)$ if any, or 0 otherwise. As we did for one-player, we assume in the following that all rotor games are tree-like and stopping.

When u_0 and ρ are fixed, this defines a zero-sum game where $\mathcal{MAX}$ and $\mathcal{MIN}$ try respectively to maximize and minimize the value of the game by choosing an appropriate strategy, respectively in $\Phi_{\mathcal{MAX}}$ and $\Phi_{\mathcal{MIN}}$. Usually, such a zero-sum game does not always have an equilibrium in pure strategies and so-called mixed (i.e., stochastic) strategies are required; this is the case in the example of Figure 4.11 where the given graph is not tree-like. However, in the case of tree-like multigraphs we prove the following theorem.

Theorem 4.6.1 (Existence of pure strategy Equilibrium)**.** *Let G be a tree-like two-player rotor game together with a starting vertex u_0. Then there are an integer value val* and two strategies ϕ^*, τ^* such that*

1. $\forall \tau \in \Phi_{\mathcal{MIN}}$, $\mathsf{val}_{\phi^*,\tau}(u_0) \geq \mathsf{val}^*$, i.e., *$\tau^*$ is optimal in the one-player game $G(\phi^*, \cdot)$,*
2. $\forall \phi \in \Phi_{\mathcal{MAX}}$, $\mathsf{val}_{\phi,\tau^*}(u_0) \leq \mathsf{val}^*$, i.e., *$\phi^*$ is optimal in the one-player game $G(\cdot, \tau^*)$.*

We call val the value of the game and the pair (ϕ^*, τ^*) is a pure strategy equilibrium.*

Furthermore val can be computed in the same time complexity as the computation of $D(\rho)$ in the zero-player game (see Theorem 4.4.3).*

This theorem is proved by following the same scheme as for the one player game: first, we consider the binary case, and then the general case follows by dichotomy. A constructive proof is given in both cases.

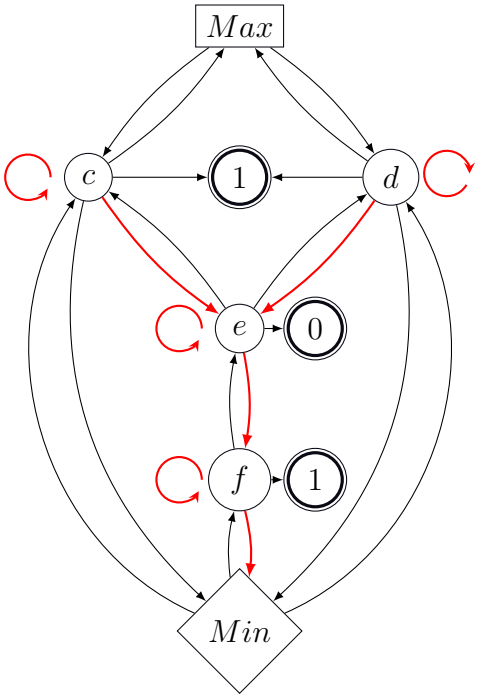


Figure 4.11: This example is a simple undirected graph where each edge is replaced by two arcs. We have $V_{\mathcal{MAX}} = \{Max\}$, $V_{\mathcal{MIN}} = \{Min\}$ and $V_r = \{c, d, e, f\}$ and S_0 is the rest of the vertices with their value written inside them. The particle starts on the vertex Max. In this game, the only optimal strategy for $\mathcal{MAX}$ when the strategy for $\mathcal{MIN}$ is the arc (Min, x) with $x \in \{c, d\}$ is (Max, x). On the other hand the only optimal strategy for $\mathcal{MIN}$ when the strategy for $\mathcal{MAX}$ is the arc (Max, c) (resp. (Max, d)) is (Min, d) (resp. (Min, c)). The situation is like the classical matching pennies game (see [69] for precise definition) where one player tries to match the strategy of the opponent whereas the other player has the opposite objective. It is known that such game does not admit a Nash equilibrium in pure strategies.

4.6.1 . Two-player Binary Rotor Game

In this subsection, we restrict the game to the case where values of sinks are binary numbers i.e., val$(s) \in \{0, 1\}$ for all $s \in S_0$.

Definition 4.6.2 (Subtree equilibrium)**.** *A pair of strategies ϕ^*, τ^* is a* subtree equilibrium *at u_0 if ϕ^* (resp., τ^*) is subtree optimal at u_0 in the one-player game $G(\cdot, \tau^*)$ (resp., $G(\phi^*, \cdot)$).*

Lemma 4.6.3. *For any $(u, v) \in \hat{A}$ and $u_0 \in V$, there is a subtree equilibrium at u_0 for the two-player game on $T_{(u,v)}$.*

Proof. This is showed by induction.

This is clearly true if $v \in S_0$.

Otherwise, let $(u, v) \in \hat{\mathcal{A}}$. Assume that there is a subtree equilibrium (ϕ^*_w, τ^*_w) for every (v, w)-subtree with $w \in \Gamma^+(v)\backslash\{u\}$. Consider the pairs of strategies ϕ_v, τ_v defined on $T_{(u,v)}$ such that $\phi_v(z) = \phi^*_w(z)$ if $z \neq v$ and $z \in T_{(v,w)}$ and $\tau_v(z) = \tau^*_w(z)$ if $z \neq v$ and $z \in T_{(v,w)}$.

Using Lemma 4.5.6:

- if $v \in V_r$ then ϕ_v (resp., τ_v) is subtree optimal in $T_{(u,v)}(\cdot, \tau_v)$ (resp., $T_{(u,v)}(\phi_v, \cdot)$);
- if $v \in V_{\mathcal{MAX}}$ then we choose $\phi_v(v)$ as in Lemma 4.5.6 so that ϕ_v is subtree optimal in $T_{u,v}(\cdot, \tau_v)$; on the other side τ_v remains subtree optimal in $T_{u,v}(\phi_v, \cdot)$;
- we proceed similarly if $v \in V_{\mathcal{MIN}}$.

□

This proves the existence of pure strategy Equilibrium as stated in Theorem 4.6.1. Using a minimax-like algorithm, we get the time complexity claimed in the theorem.

4.6.2 . Two-player Integer Rotor Game

By analogy with the one-player case, we can search the value of the game val* in a dichotomic way. Recall that it is defined as the maximal value that player $\mathcal{MAX}$ can guarantee against any strategy of $\mathcal{MIN}$. To know if $\mathcal{MAX}$ can guarantee at least value x, it suffices to solve the binary game obtained by replacing all values of sinks greater or equal than x by 1 and the other by 0. Then, the value of the binary game is 1 if and only if the value of the initial game is at least x. By iterating this, val* can be determined by solving $O(\log(|S_0|))$ binary games.

Once val* is known, we found no simple process to deduce an equilibrium from the value of the game: this remains an open problem.

4.7 . Simple Graphs

In this section we consider simple graphs i.e., graphs such that for each vertex $u \in V$ there is at most one arc with head v when $v \in \hat{\mathcal{A}}^+(u)$. We show specific results for this kind of graphs: firstly we give a formula allowing to propagate the return flow without using the routine and then we show that this helps to efficiently solve the decisional framework when values are not restricted to be binary.

We introduce some notations that will be useful in the rest of this section. Consider two arcs b, c of $\hat{\mathcal{A}}^+(u)$, in order to measure which one is reached first by successive rotor order operations starting on a particular arc $a \in \hat{\mathcal{A}}^+(u)$ we define the *distance in a rotor orbit*.

Definition 4.7.1 (Distance in a rotor orbit)**.** *Given a vertex u and $a, b \in \hat{\mathcal{A}}^+(u)$, we denote by $d_u(a, b)$ the smallest $i \geq 0$ such that $\theta_u^i(a) = b$.*

The following operator basically checks which arc between b and c is encountered first while listing the orbit of θ_u starting from a.

Definition 4.7.2. *Let a, b, c be arcs of $\hat{\mathcal{A}}^+(u)$. We define the operator B_u such that $B_u(a, b, c) = 1$ if $d_u(a, b) \leq d_u(a, c)$ and $B_u(a, b, c) = 0$ if not.*

4.7.1 . Zero-player Game

In a simple graph, by definition $|\Gamma^+(u)|$ is equal to $|\hat{\mathcal{A}}^+(u)|$. So each time that the rotor makes one full turn at vertex u, the particle travels exactly once between u and v for each $v \in \Gamma^+(u)$. Consequently, we have the following result.

Lemma 4.7.3. *If $D(\rho)(u) = (u, v)$ then all arcs (u, w) of $\hat{\mathcal{A}}^+(u)$ are visited exactly*

$$F_\rho(u, w) = \zeta_\rho(u, v) - B_u(\rho(u), (u, v), (u, w))$$

times during a maximal rotor walk starting from (ρ, u).

This allows to compute $D(\rho)(u)$ from the return flows of the outgoing arcs of u, as stated in next lemma.

Lemma 4.7.4. *Among all arcs of $\hat{\mathcal{A}}^+(u)$ which have a minimal return flow, the arc $D(\rho)(u)$ is the first one with respect to order θ_u starting at $\rho(u)$. The flow on all arcs $(u, v) \in \hat{\mathcal{A}}$ can be computed in time $O(|\Gamma^+(u)|)$.*

Proof. Assume that $D(\rho)(u) = (u, v)$. From Lemma 4.7.3, all arcs (u, w) are visited exactly $F_\rho(u, w) = \zeta_\rho(u, v) - B_u(\rho(u), (u, v), (u, w))$ times during the maximal rotor walk starting from (ρ, u). Note that, for all $w \in \Gamma^+(u)$, $B_u(\rho(u), (u, v), (u, w))$ can be computed in time $O(|\Gamma^+(u)|)$ by applying θ_u successively starting from $\rho(u)$.

Then, for all $w \in \Gamma^+(u) \setminus v$, we have:

$$\zeta_\rho(u, w) > F_\rho(u, w) = \zeta_\rho(u, v) - B_u(\rho(u), (u, v), (u, w)) \geq \zeta_\rho(u, v) - 1$$

where the first inequality comes from Lemma 4.3.9. This gives $\zeta_\rho(u, v) < 1 + \zeta_\rho(u, w)$, or, equivalently, $\zeta_\rho(u, v) \leq \zeta_\rho(u, w)$. Hence $\zeta_\rho(u, v)$ is minimal.

On the other side, if $\zeta_\rho(u, w) = \zeta_\rho(u, v)$ then $B_u(\rho(u), (u, v), (u, w)) = 1$, which means that (u, v) is the first arc with respect to order θ_u, starting at $\rho(u)$, among all arcs of $\hat{\mathcal{A}}^+(u)$ with minimal return flow.

In that process, we also show the second part of the result i.e., the flow on arc (u, w) can be computed in time $O(|\Gamma^+(u)|)$. □

Let $(u, v) \in \hat{\mathcal{A}}$ with $u \in V_0$. Consider the (u, v)-subtree and let (v, w) be the orientation of v in the Destination Forest in that subtree. From Lemma 4.7.3 we have $\zeta_\rho(u, v) = \zeta_\rho(v, w) + B_v(\rho(v), (v, u), (v, w))$. Combining this with Lemma 4.7.4, we obtain the following recursive computation of $\zeta_\rho(u, v)$.

$$\zeta_\rho(u, v) = \min_{w \in \Gamma^+(v) \setminus \{u\}} \zeta_\rho(v, w) + B_v(\rho(v), (v, u), (v, w)) \tag{4.1}$$

We then have a result similar to that of Lemma 4.4.2.

Lemma 4.7.5 (Simple Retropropagation for the Zero-player Game)**.** *Given $v \in V$, assuming the return flows of all arcs (v, w) for $w \in \Gamma^+(v)$ are known, one can compute the return flow of all arcs (z, v) with $z \in \Gamma^-(v)$ by applying Equation* (4.1) *at most twice in time $O(\min(|\Gamma^+(v)|, |\Gamma^-(v)|))$.*

Proof. For every vertex $z \in \Gamma^-(v) \setminus \Gamma^+(v)$ we set $\zeta_\rho(z, v) = 1$ (or $\zeta_\rho(z, v) = 0$ if $z \in S_0$).

As the graph is stopping, there is at least one vertex $w \in \Gamma^+(v)$ such that the return flow of (v, w) is finite. Let $(v, w_0) = D_\rho(v)$ be the first arc with respect to order θ_v starting from $\rho(v)$ such that $\zeta_\rho(v, w_0) = \min_{w \in \Gamma^+(v)} \zeta_\rho(v, w)$. Then, for any (w, v)-subtree with $w \in \Gamma^-(v)$ and $w \neq w_0$, the last outgoing arc of v when routing a particle from (ρ, v) is (v, w_0), hence by Equation (4.1), we have $\zeta_\rho(w, v) = \zeta_\rho(v, w_0) + B_v(\rho(v), (v, w), (v, w_0))$. Note that all values $B_v(\rho(v), (v, w), (v, w_0))$ can be computed in time $O(|\Gamma^+(v)|)$ by iterating θ_v from $\rho(v)$. It remains to compute $\zeta_\rho(w_0, v)$ which can be done by applying Equation (4.1) once more. Potentially, the return flows of all arcs (v, w) with $w \in \Gamma^+(v) \setminus \{w_0\}$ might be infinite. In that case, we have $\zeta_\rho(w_0, v) = +\infty$. □

From here, we have all the tools to construct our recursive algorithm.

Theorem 4.7.6 (Complete Destination Algorithm for simple graphs)**.** *The configuration $D(\rho)$ can be computed in time complexity $O(|\mathcal{V}|)$.*

Proof. We use the same algorithm as for the multigraph case (see Theorem 4.4.3), except that we replace routine IRR used for propagating the return flow by Equation (4.1), which is more efficient since it does not need a division. And Theorem 4.4.2 is replaced by Theorem 4.7.5. Since the graph is simple, we have $|\hat{\mathcal{A}}| = |\mathcal{A}| = O(|\mathcal{V}|)$ hence the result. □

4.7.2 . One-player Simple Tree-like Rotor Game

We now consider one-player rotor games when the graph is simple. For a simple tree-like rotor game, with binary values, we show that we can compute the optimal value of every starting vertex in the same time as in Theorem 4.7.6. Then, contrary to the case presented in Section 4.5.2, for a simple game with integer values, we can achieve linear complexity to compute the value of the game.

First, Lemma 4.7.7 is a direct consequence of Lemma 4.7.4. It characterizes a subset of arcs likely to be the last outgoing arc of v.

Lemma 4.7.7. *Consider a one-player simple tree-like rotor game G, a vertex $u \in V_0$ and a strategy ϕ. Then, we have $h(D(\rho,\phi)(u)) \in \operatorname{argmin}_{v\in\Gamma^+(u)}\zeta_\phi(u,v)$.*

This is true in particular for an optimal strategy of the one-player simple tree-like rotor game. Thus, we can propagate the values and return flows more easily than for the one-player tree-like rotor game.

4.7.2.1 Binary Values

Consider a one-player simple tree-like rotor game with binary values and a strategy ϕ for the player.

Lemma 4.7.8 (Optimal Values and Return Flows)**.** *Let $(u,v) \in \hat{\mathcal{A}}$. Then:*

1. *If $v \in V_r$, suppose that we know all optimal values $\mathrm{val}^*(v,w)$ and optimal return flows $\zeta^*(v,w)$ of all arcs (v,w) with $w \neq u$. From this we can compute $\mathrm{val}^*(u,v)$ and $\zeta^*(u,v)$ by:*

   $\mathrm{val}^*(u,v) = \mathrm{val}^*(v,w_f)$ *and* $\zeta^*(u,v) = \zeta^*(v,w_f)+B_v(\rho(v),(v,u),(v,w_f))$

   where $w_f = h(D(\rho,\phi)(v))$, i.e., (v,w_f) is the last outgoing arc of v.

2. *If $v \in V_{\mathcal{MAX}}$ and if there is at least one arc (v,w_1) with $w_1 \in \Gamma^+(v)\setminus\{u\}$ such that $\mathrm{val}^*(v,w_1) = 1$ and $\zeta^*(v,w_1) = \min_{w\in\Gamma^+(v)\setminus\{u\}}\zeta^*(v,w)$ then,*

$$\mathrm{val}^*(u,v) = 1 \text{ and } \zeta^*(u,v) = \begin{cases} \zeta^*(v,w_1) \text{ if } (v,u) \in \hat{\mathcal{A}}; \\ 1 \text{ otherwise}. \end{cases}$$

3. *If $v \in V_{\mathcal{MAX}}$ and if there is no arc (v,w_1) as in case 2, we have*

$$\mathrm{val}^*(u,v) = 0 \text{ and } \zeta^*(u,v) = \begin{cases} 1+\min_{w\in\Gamma^+(v)\setminus\{u\}}\zeta^*(v,w) \text{ if } (v,u) \in \hat{\mathcal{A}}; \\ 1 \text{ otherwise}. \end{cases}$$

Note that $\zeta^*(u,v)$ can be infinite.

Proof. 1. If $v \in V_r$, we use Equation (4.1).

2. Let (v,w_1) be an arc with $w_1 \in \Gamma^+(v)\setminus\{u\}$ such that $\mathrm{val}^*(v,w_1) = 1$ and $\zeta^*(v,w_1) = \min_{w\in\Gamma^+(v)\setminus\{u\}}\zeta^*(v,w)$. Lemma 4.7.3 implies that there is a strategy ϕ with $D(\rho,\phi) = (v,w_1)$, hence $\mathrm{val}^*(u,v) = 1$. Furthermore, given any strategy ϕ such that $\mathrm{val}_\phi(u,v) = 1$, and for all w such that $\mathrm{val}^*(v,w) = 1$ we have $\zeta_\phi(v,w) \geq \zeta^*(v,w) \geq \zeta^*(v,w_1)$, hence $\zeta^*(u,v) = \zeta^*(v,w_1)$.

3. Otherwise, all arcs with minimal return flows have optimal value 0. Obviously, this implies $\mathrm{val}^*(u,v) = 0$. Given any strategy ϕ, for all w such that $\mathrm{val}^*(v,w) = 0$, we have $\zeta^*(v,w) \geq \zeta_\phi(v,w)$ hence $\min_{w\in\Gamma^+(v)\setminus\{u\}} \zeta^*(v,w) \geq \min_{w\in\Gamma^+(v)\setminus\{u\}} \zeta_\phi(v,w)$. Furthermore, if $(v,u) \in \hat{\mathcal{A}}$, there is a strategy (namely choosing (v,u) as the starting configuration of v) such that (v,u) is visited $\min_{w\in\Gamma^+(v)\setminus\{u\}} \zeta^*(v,w) + 1$ times. Hence, we have $\zeta^*(u,v) = 1 + \min_{w\in\Gamma^+(v)\setminus\{u\}} \zeta^*(v,w)$.

□

By analogy with Lemma 4.5.6, we can construct a recursive linear algorithm to compute the optimal value of u_0. But, contrary to the one-player tree-like rotor game on multigraphs, as the graph is simple, we can also compute in linear time the return flows and values of all arcs of $\hat{\mathcal{A}}$ that are not directed from u_0 towards the leaves. The reason we should do this is that, if we know all return flows of all arcs, in the end we can compute optimal values and strategies for all starting vertices simultaneously. Indeed, by using conjointly Lemma 4.7.5 and Lemma 4.7.8 this can be done in at most $2|\mathcal{A}^+(u)|$ steps for a given vertex u. This result is properly stated in the following Lemma.

Lemma 4.7.9 (Optimal Retropropagation)**.** *Given a vertex $v \in V$ and assuming the return flows and values of all arcs (v,w) for $w \in \Gamma^+(v)$ are known, one can compute all optimal return flows and all optimal values of arcs (u,v) with $u \in \Gamma^-(v)$ by applying at most twice Equation* (4.1)*.*

Proof. If $v \in V_r$, this is exactly Lemma 4.7.5 concerning the zero-player case. So we focus on the case where $v \in V_{\mathcal{MAX}}$.

Let $\Gamma^+_{\min}(v) \subset \Gamma^+(v)$ be the set of w in $\mathrm{argmin}_{w\in\Gamma^+(v)}\zeta^*(v,w)$. From Lemma 4.7.7, we have that, no matter the strategy ϕ chosen on v, $h(D(\rho,\phi)(v)) \in \Gamma^+_{\min}(v)$. From here we distinct two cases.

- Either there is a vertex $w_1 \in \Gamma^+_{\min}(v)$ such that $\mathrm{val}^*(v,w_1) = 1$ (there might be several such vertices). In this case, Lemma 4.7.8 (case 2) states that for all arcs (u,v) with $u \neq w_1$ we have $\mathrm{val}^*(u,v) = 1$ and $\zeta^*(u,v) = \zeta^*(v,w_1)$ if there is an arc (v,u) and $\zeta^*(u,v) = 1$ if not. If $(w_1,v) \in \hat{\mathcal{A}}$, it remains to consider the case of (w_1,v), which can be done by applying a second time Lemma 4.7.8 (case 2) with $u = w_1$.

- If there is no such vertex $w_1 \in \Gamma^+_{\min}(v)$ such that $\mathrm{val}^*(v,w_1) = 1$, choose any vertex $w_0 \in \Gamma^+_{\min}(v)$. By using Lemma 4.7.8 (case 3) for all arcs (u,v) with $u \neq w_0$ we have that $\mathrm{val}^*(u,v) = 0$ and $\zeta^*(u,v) = \zeta^*(v,w_0) + 1$ if there is an arc (v,u), and $\zeta^*(u,v) = 1$ if not. As for the previous case, if $(w_0,v) \in \hat{\mathcal{A}}$, it remains to consider the case of (w_0,v), which can be done by applying a second time Lemma 4.7.8 (case 3) with $u = w_0$.

□

This allows us to compute the optimal values of all vertices in linear time.

Theorem 4.7.10. *Given a one-player binary tree-like rotor game on a simple graph, we can compute the optimal values of all vertices in time complexity $O(|V|)$.*

Proof. We consider the arcs in the same order than in Theorem 4.4.3. For all arcs directed from u_0 towards the leaves, we use Lemma 4.7.8. This is done in $O(|\hat{\mathcal{A}}|)$ comparisons. Then, for all arcs directed from the leaves towards u_0 we use Theorem 4.7.9 which is done in $O(|\hat{\mathcal{A}}|)$ comparisons. Then, we just need to compute the optimal value of each vertex as in the proof of Theorem 4.5.7. Which is done in $O(|\mathcal{A}|)$ comparisons as well. □

Remark. *This procedure gives us the optimal value of the game simultaneously for every starting vertex, in overall linear time. However, as noted before (see Figure 4.8), the optimal strategy depends on the starting vertex.*

This extension is also valid for the two-player binary variant on a simple graph as the subtree optimal equilibrium is preserved for the same reason as in Section 4.6.1.

4.7.2.2 One-player Simple Integer Rotor Game

This last subsection introduces a notion of *access flow* that measures the potential access to an arc for a rotor walk starting at u_0. It allows us to compute the optimal value of the game in linear time for the integer case on a simple graph, in contrast to the general case where we had to use a bisection algorithm (Section 4.5.2).

Lemma 4.7.11 (Strategy ϕ_{max})**.** *Given a vertex u_0, there is a strategy $\phi_{max} \in \Phi_{\mathcal{MAX}}$ such that, for all arc $(u_0, v) \in \hat{\mathcal{A}}$,*

$$\zeta_{\phi_{max}}(u_0, v) = \max_{\phi \in \Phi_{\mathcal{MAX}}} \zeta_{\phi}(u_0, v),$$

namely any strategy that chooses to direct arcs towards u_0 when possible.

Proof. We prove recursively that the strategy ϕ_{max} described above is maximal, starting from leaves and going back to u_0. Let $(u, v) \in \hat{\mathcal{A}}$ such that (u, v) is directed towards the leaves, suppose that all return flows of arcs $(v, w) \in \hat{\mathcal{A}}$ with $w \in \Gamma^+(v) \setminus \{u\}$ are maximal. If $v \in V_r$, $\zeta_\phi(u, v)$ is maximal if the return flow of $D(\rho, \phi)(v)$ is maximal among all strategies ϕ. By choosing $\phi_{max}(v) = (v, u)$, we have $B_v(\phi(v), (v, u), (v, w)) = 1$ for any $w \in \Gamma^+(v) \setminus \{u\}$. At the same time, $\phi(v)$ is directed towards u_0. From Equation (4.1), we then have that, no matter the value of the return flow of the arcs (v, w), as long as it is maximal, the return flow of (u, v) is maximal. If there is no arc (v, u), the choice does not matter as $\zeta_\phi(u, v) = 1$ for any strategy $\phi \in \Phi_{\mathcal{MAX}}$. □

Definition 4.7.12 (Access flow acc(u,v))**.** *For every arc $(u,v) \in \hat{\mathcal{A}}$, we denote by acc$(u,v)$ the value such that, if we remove all outgoing arcs of v and we add an arc (v,u), acc(u,v) is the maximal number of times arc (u,v) is visited in the maximal rotor walk starting from $((\rho,\phi),u_0)$ for all strategies $\phi \in \Phi_{\mathcal{MAX}}$. Note that acc$(u,v)$ might be infinite in the case where all sinks are in $T_{(u,v)}$.*

In particular, acc(u,v) is positive if and only if there exists a strategy ϕ such that v is visited at least once in the maximal rotor walk starting from $((\rho,\phi),u_0)$.

A sink s is said to be *reachable* if, for $(u,s) \in \hat{\mathcal{A}}$, acc$(u,s)$ is positive. Hence, to solve our problem we just need to find the sink $s_{\max}$ of maximal value among the reachable sinks.

From here we present the linear process that allows to compute the access flow of all arcs with head in S_0.

Lemma 4.7.13 (Access flow around u_0)**.** *Let ϕ_{max} be the strategy detailed above. For any $v_i \in \Gamma^+(u_0)$ we have:*

$$\textit{acc}(u_0,v_i) = \begin{cases} \min_{v_j, j\neq i} \zeta_{\phi_{max}}(u_0,v_j) - B_{u_0}(\rho(u_0),(u_0,v_j),(u_0,v_i)) & \text{if } u_0 \in V_r \\ \min_{v_j, j\neq i} \zeta_{\phi_{max}}(v_0,v_j) & \text{if } u_0 \in V_{\mathcal{MAX}} \end{cases}$$

Proof. To maximize the number of times (u_0,v_i) is visited in the definition of acc(u_0,v_i), we need to maximize the return flows of all arcs (u_0,v_j). All these arcs are directed from u_0 towards leaves, so the return flow of these arcs under strategy ϕ_{max} is maximum among all strategies. Both cases derive from Equation (4.1), but in the second one, u_0 is initially directed towards v_i. □

Similarly to the return flow, we give a recursive equation that computes the access flow on all arcs directed from u_0 towards the leaves.

Lemma 4.7.14 (Access flow Propagation)**.** *Let $(u,v) \in \hat{\mathcal{A}}$, let $w_0,..,w_k$ be the vertices of $\Gamma^+(v) \setminus u$. If we know acc(u,v), then, for any couple (v,w_i) we have:*

$$\textit{acc}(v,w_i) = \min \begin{cases} \min_{w_j, j\neq i} \zeta_{\phi_{max}}(v,w_j) - B_v(\rho(v),(v,w_j),(v,w_i)); \\ \textit{acc}(u,v) - B_v(\rho(v),(v,u),(v,w_i)) \end{cases}$$

If $v \in V_{\mathcal{MAX}}$, we have:

$$\textit{acc}(v,w_i) = \min \begin{cases} \min_{w_j, j\neq i} \zeta_{\phi_{max}}(v,w_j); \\ \textit{acc}(u,v) \end{cases}$$

as the player can always choose $\phi(v) = (v,w_i)$.

Once again, using this formula to compute all access flows might take more than a linear time, so we give a property similar to Lemma 4.7.5 allowing to compute all access flows in linear time.

Lemma 4.7.15 (Access flow Computation). *Let $(u, v) \in \hat{\mathcal{A}}$ and $w_0, .., w_k$ be the vertices of $\Gamma^+(v) \setminus u$, and assume that acc(u, v) is known. Then acc(v, w_i) can be determined for all $i \in \{0 \dots k\}$ by computing acc(v, w_i) for only two values of i.*

Proof. Same proof than for Lemma 4.7.5 if vertex v belongs to V_r. If $v \in V_{\mathcal{MAX}}$, we also use Lemma 4.7.5 but with $B_v(\phi(v), (v, u), (v, w_i)) = 0$ for all $w_i \in \Gamma^+(v) \setminus \{u\}$. □

Theorem 4.7.16. *One can compute the value of a one-player tree-like rotor game with arbitrary integer values in linear time $O(|V|)$ for a given starting vertex u_0.*

Proof. Consider the arcs in the same order than in Theorem 4.4.3. For all arcs directed from u_0 towards the leaves, we compute the maximal return flows recursively from the leaves (i.e., constructing strategy ϕ_{max}). Then, for all arcs with tail u_0 we use Lemma 4.7.13 and finally we use Lemma 4.7.15 to compute the access flow on all arcs directed towards the leaves. Each arc is considered at most twice so the complexity of these steps sums up to $O(|\hat{\mathcal{A}}|)$. After that, the value of the game is the reachable sink with maximal value which is computed in $O(|V|)$. As the graph is tree-like and simple we have $|\hat{\mathcal{A}}| \leq 2|V|$ hence our complexity result. □

As a concluding example, let us apply this Theorem to the example of Figure 4.12 to compute the correct value of the game.

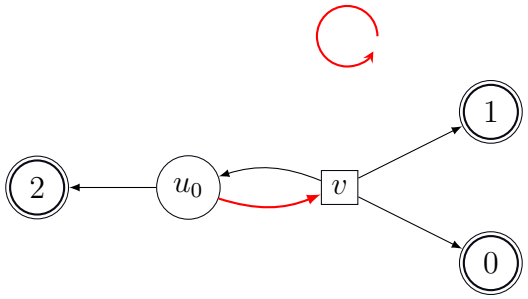


Figure 4.12: Example of a simple integer tree-like rotor game. Here, $v \in V_{\mathcal{MAX}}$ and $u_0 \in V_r$. All other vertices belong to S_0 and their value is written inside them. The initial configuration of u_0 is the red arc in dashes.

First, compute the return flows of arcs $(u_0, 2)$ and (u_0, v) under strategy $\phi_{\max}$. We have $\zeta_{\phi_{\max}}(u_0, 2) = 1$ and $\zeta_{\phi_{\max}}(u_0, v) = 2$. From Theorem 4.7.13, we have acc$(u_0, 2) = 1$ as $B_{u_0}(\rho(u_0), (u_0, v), (u_0, 2)) = 1$. From the same formula we have acc$(u_0, v) = 1$. And by Theorem 4.7.15 we have acc$(v, 1) =$ acc$(v, 0) = 1$. All sinks are reachable, and 2 is the maximal value among them, hence val$^*(v_0) = 2$.

Conclusion and Future Works

This chapter presented polynomial algorithms to solve **SP-ARRIVAL** on treelike-multigraphs but also its decisional variants with one and two players. These algorithms rely on the notion of *return flow* that can only be used in the underlying graph is a tree.

Concerning **SP-ARRIVAL**, one remaining fundamental question is to determine whether a polynomial algorithm exists to solve the zero-player game. Similarly, problems such as *simple stochastic games, parity games* and *mean-payoff games* are also in NP $\cap$ co-NP, and no polynomial algorithm is known to solve them (see [47]). For those different problems, considering sub-classes of graphs where we can find polynomial algorithms is a fruitful approach (see [4] and [5]).

Thus, we would like to study more general classes of graphs. To begin with, even graphs that are well-studied in terms of the *sandpile group* such that ladders or grids remain an open problem for **SP-ARRIVAL**. The problem of finding the destination of multiple particles (e.g., **MP-ARRIVAL**) at the same time is also a significant open problem in nearly all cases except the path graph.

Finally, another natural extension of our algorithm would be to define and study an adequate notion of graphs with bounded width to generalize the Tree-like multigraph case.

Algorithme 3 : Optimal Strategy Computation

input : (u, v) is an arc of $\hat{\mathcal{A}}$; $\zeta : \Gamma^+(v) \setminus \{u\} \to \mathbb{Z} \cup +\infty$ contains the optimal return flows of arcs $(v, w) \in \hat{\mathcal{A}}$; $val : \Gamma^+(v) \setminus \{u\} \to \{0, 1\}$ contains the optimal values of these arcs.

output : an element of $\mathcal{A}^+(v)$ (optimal strategy $\sigma^*(v)$); a positive integer ($\zeta^*(u, v)$) and a binary integer (val$^*(u, v)$))

For $w \in \Gamma^+(v)$, let $Q(w)$ be the quotient of the euclidean division of $\zeta(w)$ by $|h^{-1}(w)|$; let $q_{\min}$ be the minimum value of $Q(w)$ for all w; let $R(w)$ be $\zeta(w) - (q_{\min} * |h^{-1}(w)|)$; let $a_s = a_e = a_0$ with a_0 being any arc of $\mathcal{A}^+(v)$, let $f_0, f_1, f_u, value$ be integers initialized to 0. Let b_0, b_1 be two arcs initialized to a_0. Let *last* be a boolean with initial value False.

if $r(w) = +\infty, \forall w \in \Gamma^+(v) \backslash \{u\}$ **then**
    return $b_0, +\infty, 0$
repeat
    if $h(a_e) = u$ **then**
        $f_u \mathrel{+}= 1$
        $a_e \leftarrow \theta_v(a_e)$
    else
        $R(h(a_e)) \mathrel{-}= 1$
        if $R(h(a_e)) = 0$ **then**
            if $val(a_e) = 0$ **then**
                `/* Keep track of the maximal return with value 0 */`
                if $f_0 < f_u$ **then**
                    $f_0 \leftarrow f_u$
                    $b_0 \leftarrow a_s$
            else
                if $f_1 > f_u$ **then**
                    `/* Keep track of the minimal with value 1 and recall the fact that there exist a strategy with value 1 */`
                    $f_1 \leftarrow f_u$
                    $b_1 \leftarrow a_s$
                $value \leftarrow 1$
            if $h(a_s) = u$ **then**
                $f_u \mathrel{-}= 1$
            else
                $R(h(a_s)) \mathrel{+}= 1$
            $a_s \leftarrow \theta_v(a_s)$
            if $\theta_v(a_s) = a_0$ **then**
                last $\leftarrow$ True
        else
            $a_e \leftarrow \theta_v(a_e)$
until $a_s = a_0$ *and last=True*
if $value = 1$ **then**
    return $b_1, f_1 + q_{min} * |h^{-1}(u)| + 1, 1$
else
    return $b_0, f_0 + q_{min} * |h^{-1}(u)| + 1, 0$

5 - Conclusion

This work represents an in-depth exploration of the rotor-routing model ([15]) within the context of reachability problems. We extensively investigate various variants of the ARRIVAL problem, as introduced in [31], as well as a diverse range of problems that involve a combination of rotor-routing, sandpiles, and the dollar game. Additionally, we delve into the study of the ARRIVAL problem on a specific class of graphs, which we refer to as treelike-multigraphs.

In Chapter 2, our objective was to consolidate the existing literature on the ARRIVAL problem and reachability problems within the rotor-routing model framework. This approach resulted in the creation of a unified framework for rotor-routing models. This framework encompasses not only rotor walks and rotor routing sequences but also negative routing operations (i.e., antiparticles) and legal and non-legal routing operations. We successfully extended most of the known results on rotor-routing models to this unified framework, which also forms the basis of our work in Chapter 3.

Additionally, in Chapter 3, we formulated various reachability problems within this unified framework, considering two key parameters: legal or non-legal routing sequences and stopping or strongly connected graphs. We compared the complexity of various reachability problems to that of the ARRIVAL problem. One significant contribution of this document has been to demonstrate that the ARRIVAL problem with multiple particles is equivalent to the problem with non-legal routing operations. This finding indicates that the complexity of the ARRIVAL problem is primarily influenced by the rotor-routing mechanism and graph topology, rather than the legality constraint.

This final result strengthens our approach to studying the ARRIVAL problem on specific graph topologies in Chapter 4 for a more in-depth understanding. In particular, we provided a polynomial algorithm to solve the ARRIVAL problem on treelike-multigraphs. Additionally, we utilized these results to solve deterministic analogues of Markov Decision Processes and Stochastic Games on treelike-multigraphs, thereby enhancing the connections between random and rotor walks.

Following the example of $(*, \sigma) \sim (\rho', \sigma')$, we conjectured in Chapter 3 that the problem $(*, \sigma) \to (\rho', \sigma')$ is equivalent to the ARRIVAL problem with multiple particles. If this property were to hold true, it would imply complete symmetry in the ARRIVAL problem. Essentially, determining the exit sink based on the initial rotor particle configuration would be equivalent to finding the initial rotor configuration based on the final rotor particle configuration, and vice versa. We also noted that the reachability problems presented in Table 3.2 could be approached from another perspective by considering equivalence classes instead of individual configurations. For example, consider the problem $(\rho, \sigma) \sim (*, \sigma_0)$ in the linear (i.e., non-legal) case. Given a rotor particle configuration (ρ, σ), for any rotor

particle configuration (ρ', σ') such that $\rho \sim \rho'$ and $\sigma \sim \sigma'$, we have $(\rho, \sigma) \sim (\rho', \sigma')$. This holds true for σ_0 and any particle configuration σ_0' such that $\sigma_0 \sim \sigma_0'$. It implies that the problem of determining whether there exists ρ'' with $\rho \sim \rho''$, σ'' with $\sigma \sim \sigma''$, and σ_0'' with $\sigma_0 \sim \sigma_0''$ such that $(\rho'', \sigma'') \sim (*, \sigma_0'')$ is equivalent to the problem $(\rho, \sigma) \sim (*, \sigma_0)$. However, in the legal case, there is not always a legal routing sequence between elements of the same rotor class or the same particle configuration class, which makes this problem more challenging. Investigating such problems appears to be a promising avenue for gaining a better understanding of rotor classes and particle configuration classes (e.g., Sandpiles group).

In Chapter 4, we posed several open questions. We were unable to provide a polynomial algorithm for any variant of the ARRIVAL problem on general digraphs. Specifically, as long as the graph contains a vertex-disjoint circuit of length more than three, the tool we introduced for solving the ARRIVAL problem on treelike-multigraphs, namely the return flow, is no longer sufficient. Nevertheless, it appears that this approach can potentially be generalized to graphs of the form illustrated in Figure 5.1.

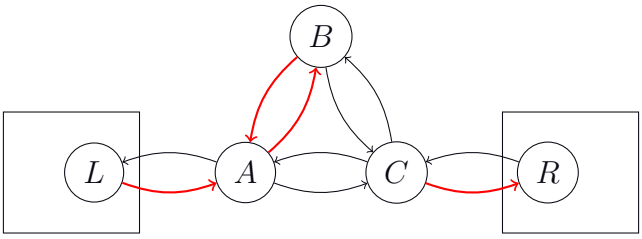


Figure 5.1: Graph with a triangle ABC and where the rest of the vertices can be partitioned in two disjoints sets, one containing L and one containing R.

Considering that the set A, B, C does not contain a sink vertex, we treat it as a unique vertex ABC with a 'complex' rotor order. To determine the rotor order of vertex ABC based on the particle's initial position, one can compute it by observing successive routing operations. For instance, let us assume the particle first enters A, B, C via C. It then exits A, B, C through C. This implies that if it returns to A, B, C again, it will necessarily enter through C. In that case, it will exit A, B, C via A. If the particle returns once more, it will enter A, B, C via A and exit via C. A check reveals that the rotor configuration of A, B, C aligns with that shown in Figure 5.1.

Therefore, the 'complex' rotor order θ_{ABC} of ABC with the particle initially entering via C and having rotor configuration ρ is such that $h(\rho(ABC)) = R, h(\theta_{ABC}(\rho(ABC))) = L, h(\theta_{ABC}^2(\rho(ABC))) = R$. Consequently, the problem becomes equivalent to **SP-ARRIVAL**. This underscores the promising nature of investigating the computation of such rotor orders as a field of study.

As mentioned in this document, we successfully solved **MP-ARRIVAL** (i.e., the ARRIVAL problem with multiple particles) comprehensively on path graphs,

leveraging our knowledge of the Sandpiles group associated with these graphs. The understanding of the Sandpiles group of a graph is a crucial prerequisite for solving **MP-ARRIVAL** for that specific graph. Consequently, a promising avenue for research involves the computation of the Sandpiles group for various graphs.

We are aware that this field is currently under active investigation, and we anticipate that future results in this area will contribute significantly to extending our comprehension of the ARRIVAL problem.

Bibliography

[1] Carlos A Alfaro, Carlos E Valencia, and Marcos C Vargas. "Computing sandpile configurations using integer linear programming". In: *Chaos, Solitons & Fractals* 170 (2023), p. 113356.

[2] Joshua Ani et al. "Trains, games, and complexity: 0/1/2-player motion planning through input/output gadgets". In: *Theoretical Computer Science* (2023), p. 113945.

[3] David Auger, Pierre Coucheney, and Loric Duhazé. "Polynomial Time Algorithm for ARRIVAL on Tree-Like Multigraphs". In: *47th International Symposium on Mathematical Foundations of Computer Science*. 2022, p. 1.

[4] David Auger, Pierre Coucheney, and Yann Strozecki. "Finding optimal strategies of almost acyclic simple stochastic games". In: *International Conference on Theory and Applications of Models of Computation*. Springer. 2014, pp. 67–85.

[5] David Auger, Pierre Coucheney, and Yann Strozecki. "Solving Simple Stochastic Games with Few Random Nodes Faster Using Bland's Rule". In: *36th International Symposium on Theoretical Aspects of Computer Science*. 2019.

[6] David Auger et al. "Generalized ARRIVAL Problem for Rotor Walks in Path Multigraphs". In: (2023). arXiv: 2307.01897 [cs.DM].

[7] László Babai and Everlin Toumpakari. "A structure theory of the sandpile monoid for directed graphs". In: *Journal of Combinatorics, to appear* (2010).

[8] Spencer Backman and Sam Hopkins. "Fourientations and the Tutte polynomial". In: *Research in the Mathematical Sciences* 4.1 (2017), p. 18.

[9] Spencer Christopher Foster Backman. "Combinatorial divisor theory for graphs". PhD thesis. Georgia Institute of Technology, 2014.

[10] Per Bak, Chao Tang, and Kurt Wiesenfeld. "Self-organized criticality: An explanation of the 1/f noise". In: *Physical review letters* 59.4 (1987), p. 381.

[11] Henry G Baker. *Rabin's proof of the undecidability of the reachability set inclusion problem of vector addition systems*. Massachusetts Institute of Technology, Project MAC, 1973.

[12] Matthew Baker and Farbod Shokrieh. "Chip-firing games, potential theory on graphs, and spanning trees". In: *Journal of Combinatorial Theory, Series A* 120.1 (2013), pp. 164–182.

[13] Ravindra B Bapat. "The Laplacian matrix of a graph". In: *Mathematics Student-India* 65.1 (1996), pp. 214–223.

[14] Norman L Biggs. "Chip-firing and the critical group of a graph". In: *Journal of Algebraic Combinatorics* 9.1 (1999), pp. 25–45.

[15] Anders Björner and László Lovász. "Chip-firing games on directed graphs". In: *Journal of algebraic combinatorics* 1.4 (1992), pp. 305–328.

[16] Anders Björner, László Lovász, and Peter W Shor. "Chip-firing games on graphs". In: *European Journal of Combinatorics* 12.4 (1991), pp. 283–291.

[17] Richard P Brent and Paul Zimmermann. *Modern computer arithmetic*. Vol. 18. Cambridge University Press, 2010.

[18] Alberto Caprara. "Sorting permutations by reversals and Eulerian cycle decompositions". In: *SIAM journal on discrete mathematics* 12.1 (1999), pp. 91–110.

[19] Melody Chan, Thomas Church, and Joshua A Grochow. "Rotor-routing and spanning trees on planar graphs". In: *International Mathematics Research Notices* 2015.11 (2015), pp. 3225–3244.

[20] Denis Chebikin and Pavlo Pylyavskyy. "A family of bijections between G-parking functions and spanning trees". In: *Journal of Combinatorial Theory, Series A* 110.1 (2005), pp. 31–41.

[21] William Chen and Travis Schedler. "Concrete and abstract structure of the sandpile group for thick trees with loops". In: *arXiv preprint math/0701381* (2007).

[22] Stephen A Cook. "The complexity of theorem-proving procedures". In: *Proceedings of the third annual ACM symposium on Theory of computing*. 1971, pp. 151–158.

[23] Stephen A Cook and Robert A Reckhow. "Time-bounded random access machines". In: *Proceedings of the fourth annual ACM symposium on Theory of computing*. 1972, pp. 73–80.

[24] Robert Cori and Dominique Rossin. "On the sandpile group of dual graphs". In: *European Journal of Combinatorics* 21.4 (2000), pp. 447–459.

[25] Scott Corry and David Perkinson. *Divisors and Sandpiles: An Introduction to Chip-Firing*. Vol. 114. American Mathematical Soc., 2018.

[26] Vijay D'silva, Daniel Kroening, and Georg Weissenbacher. "A survey of automated techniques for formal software verification". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.7 (2008), pp. 1165–1178.

[27] Deepak Dhar. "Self-organized critical state of sandpile automaton models". In: *Physical Review Letters* 64.14 (1990), p. 1613.

[28] Deepak Dhar. "The abelian sandpile and related models". In: *Physica A: Statistical Mechanics and its applications* 263.1-4 (1999), pp. 4–25.

[29] Deepak Dhar. "Theoretical studies of self-organized criticality". In: *Physica A: Statistical Mechanics and its Applications* 369.1 (2006), pp. 29–70.

[30] Deepak Dhar et al. "Algebraic aspects of abelian sandpile models". In: *Journal of physics A: mathematical and general* 28.4 (1995), p. 805.

[31] Jérôme Dohrau et al. "ARRIVAL: A zero-player graph game in NP ∩ coNP". In: *A journey through discrete mathematics*. Springer, 2017, pp. 367–374.

[32] Noah Doman. "The Identity of the Abelian Sandpile Group". PhD thesis. 2020.

[33] Ioana Dumitriu, Prasad Tetali, and Peter Winkler. "On playing golf with two balls". In: *SIAM Journal on Discrete Mathematics* 16.4 (2003), pp. 604–615.

[34] Matthew Farrell and Lionel Levine. "CoEulerian graphs". In: *Proceedings of the American Mathematical Society* 144.7 (2016), pp. 2847–2860.

[35] John Fearnley et al. "Unique end of potential line". In: *Journal of Computer and System Sciences* 114 (2020), pp. 1–35.

[36] Jérémy Fersula, Camille Noûs, and Kévin Perrot. "Sandpile toppling on Penrose tilings: identity and isotropic dynamics". In: *Automata and Complexity*. Springer, 2022, pp. 117–143.

[37] Tobias Friedrich and Thomas Sauerwald. "The cover time of deterministic random walks". In: *International Computing and Combinatorics Conference*. Springer. 2010, pp. 130–139.

[38] Andrei Gabrielov. "Abelian avalanches and Tutte polynomials". In: *Physica A: Statistical Mechanics and its Applications* 195.1-2 (1993), pp. 253–274.

[39] Bernd Gärtner, Sebastian Haslebacher, and Hung P. Hoang. "A Subexponential Algorithm for ARRIVAL". In: *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*. Vol. 198. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 69:1–69:14.

[40] Bernd Gärtner et al. "ARRIVAL: Next Stop in CLS". In: *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2018.

[41] Giuliano Pezzolo Giacaglia et al. "Local-to-global principles for the hitting sequence of a rotor walk". In: *the electronic journal of combinatorics* (2012), P5–P5.

[42] Samir Gokarn and Thyagaraj S Kuthambalayan. "Analysis of challenges inhibiting the reduction of waste in food supply chain". In: *Journal of cleaner production* 168 (2017), pp. 595–604.

[43] Eric Goles and Maurice Margenstern. "Universality of the chip-firing game". In: *Theoretical Computer Science* 172.1-2 (1997), pp. 121–134.

[44] Martin Grötschel, László Lovász, and Alexander Schrijver. "Geometric Algorithms and Combinatorial Optimization". In: ().

[45] Sara K. Grumbacher et al. "Self-organized criticality: An experiment with sandpiles". In: *American Journal of Physics* 61.4 (1993), pp. 329–335.

[46] Michael Hack. "Decision problems for Petri nets and vector addition systems". In: (1975).

[47] Nir Halman. "Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems". In: *Algorithmica* 49.1 (2007), pp. 37–50.

[48] David Harvey and Joris Van Der Hoeven. "Integer multiplication in time O(n log n)". In: *Annals of Mathematics* 193.2 (2021), pp. 563–617.

[49] Alexander E Holroyd and James Propp. "Rotor walks and Markov chains". In: *Algorithmic probability and combinatorics* 520 (2010), pp. 105–126.

[50] Alexander E Holroyd et al. "Chip-firing and rotor-routing on directed graphs". In: *In and out of equilibrium 2*. Springer, 2008, pp. 331–364.

[51] Sam Hopkins, Thomas McConville, and James Propp. "Sorting via Chip-Firing". In: *The Electronic Journal of Combinatorics* 24.3 (2017), P3–13.

[52] Bálint Hujter, Viktor Kiss, and Lilla Tóthmérész. "On the complexity of the chip-firing reachability problem". In: *Proceedings of the American Mathematical Society* 145.8 (2017), pp. 3343–3356.

[53] Eli Timothy Johnson. "The Sandpile Group on a Hexagonal Grid". PhD thesis. 2019.

[54] Kyle D Julian and Mykel J Kochenderfer. "Reachability analysis for neural network aircraft collision avoidance systems". In: *Journal of Guidance, Control, and Dynamics* 44.6 (2021), pp. 1132–1142.

[55] CS Karthik. "Did the train reach its destination: The complexity of finding a witness". In: *Information Processing Letters* 121 (2017), pp. 17–21.

[56] Itamar Landau and Lionel Levine. "The rotor–router model on regular trees". In: *Journal of Combinatorial Theory, Series A* 116.2 (2009), pp. 421–433.

[57] Yvan Le Borgne and Dominique Rossin. "On the identity of the sandpile group". In: *Discrete mathematics* 256.3 (2002), pp. 775–790.

[58] Jan van Leeuwen. "A Partial Solution to the Reachability-Problem for Vector-Addition Systems". In: *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*. 1974, pp. 303–309.

[59] Lionel Levine. "Sandpile groups and spanning trees of directed line graphs". In: *Journal of Combinatorial Theory, Series A* 118.2 (2011), pp. 350–364.

[60] Lionel Levine. "The sandpile group of a tree". In: *European Journal of Combinatorics* 30.4 (2009), pp. 1026–1035.

[61] Rupert Li and James Propp. "A greedy chip-firing game". In: *Random Structures & Algorithms* 62.3 (2023), pp. 645–666.

[62] Criel Merino López. "Chip firing and the Tutte polynomial". In: *Annals of Combinatorics* 1.1 (1997), pp. 253–259.

[63] Graham Manuell. "A simple lower bound for ARRIVAL". In: *arXiv preprint arXiv:2108.06273* (2021).

[64] Ernst W Mayr. "An algorithm for the general Petri net reachability problem". In: *Proceedings of the thirteenth annual ACM symposium on Theory of computing*. 1981, pp. 238–246.

[65] Hagar Mosaad. “ARRIVAL: A zero-player graph game”. PhD thesis. 2017.

[66] BO Nash. “Reachability problems in vector addition systems”. In: *The American Mathematical Monthly* 80.3 (1973), pp. 292–295.

[67] Jia Ning et al. “Journal of Rail Transport Planning & Management”. In: *Journal of Rail Transport Planning & Management* 23 (2022), p. 100333.

[68] Esko Nuutila and Eljas Soisalon-Soininen. “On finding the strongly connected components in a directed graph”. In: *Information processing letters* 49.1 (1994), pp. 9–14.

[69] Martin J Osborne et al. *An introduction to game theory*. Vol. 3. 3. Oxford university press New York, 2004.

[70] Jim Pitman and Wenpin Tang. “Tree formulas, mean first passage times and Kemeny’s constant of a Markov chain”. In: *Bernoulli* 24.3 (Aug. 2018). doi: 10.3150/16-bej916. url: https://doi.org/10.3150%2F16-bej916.

[71] AM Povolotsky, VB Priezzhev, and RR Shcherbakov. “Dynamics of Eulerian walkers”. In: *Physical review E* 58.5 (1998), p. 5449.

[72] Vyatcheslav B Priezzhev et al. “Eulerian walkers as a model of self-organized criticality”. In: *Physical Review Letters* 77.25 (1996), p. 5079.

[73] James Propp. “Lattice structure for orientations of graphs”. In: *arXiv preprint math/0209005* (2002).

[74] Martin L Puterman. “Markov decision processes”. In: *Handbooks in operations research and management science* 2 (1990), pp. 331–434.

[75] Yuval Rabani, Alistair Sinclair, and Rolf Wanka. “Local divergence of Markov chains and the analysis of iterative load-balancing schemes”. In: *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*. IEEE. 1998, pp. 694–703.

[76] George S Sacerdote and Richard L Tenney. “The decidability of the reachability problem for vector addition systems (preliminary version)”. In: *Proceedings of the ninth annual ACM symposium on Theory of computing*. 1977, pp. 61–76.

[77] Rahul Savani et al. “Reachability Switching Games”. In: *Logical Methods in Computer Science* 17 (2021).

[78] Paul D. Seymour. “Packing directed circuits fractionally”. In: *Combinatorica* 15.2 (1995), pp. 281–288.

[79] Lloyd S Shapley. “Stochastic games”. In: *Proceedings of the national academy of sciences* 39.10 (1953), pp. 1095–1100.

[80] Lilla Tóthmérész. “Algorithmic aspects of rotor-routing and the notion of linear equivalence”. In: *Discrete Applied Mathematics* 236 (2018), pp. 428–437.

[81] Lilla Tóthmérész. “Rotor-routing reachability is easy, chip-firing reachability is hard”. In: *European Journal of Combinatorics* 101 (2022), p. 103466.

[82] Evelin Toumpakari. “On the sandpile group of regular trees”. In: *European Journal of Combinatorics* 28.3 (2007), pp. 822–842.

[83] Evelin Christiana Toumpakari. “On the Abelian sandpile model.” In: (2005).

[84] Geoffrey G Xie et al. “On static reachability analysis of IP networks”. In: *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.* Vol. 3. 2005, pp. 2170–2183.

[85] Vladimir Yanovski, Israel A Wagner, and Alfred M Bruckstein. “A distributed ant algorithm for protect efficiently patrolling a network”. In: *Algorithmica* 37.3 (2003), pp. 165–186.

Appendix - Complexity Class Syllabus

This section provides definitions and insights into the various *complexity classes* discussed in this document. In computational complexity, problems are categorized into *complexity classes*, and in this context, the computational complexity depends on graph parameters, such as the number of vertices or arcs in the graph.

- The **P** complexity class includes all decision problems solvable by a deterministic Turing machine in polynomial time. When we state that a problem is "solvable in polynomial time," it exists a deterministic Turing machine to solve this problem and whose computational time can be bounded by a polynomial function of the number of arcs.

- The **NP** complexity class, encompassing **P**, is widely conjectured to have a distinct status from **P** (**P**$\neq$**NP**). In this document, we adhere to the definition that **NP** includes decision problems whose solution is certifiable in polynomial time by a deterministic Turing machine, referred to as a certifier. This implies that, for any instance I with a "yes" answer, there exists a certifier that checks in polynomial time whether a certificate C (which is polynomial in the size of the entries) proves the answer to be "yes". Hence, the problem of deciding whether C demonstrates that the answer is "yes" or not falls within the **P** complexity class. It is important to note that, given a certifier, there may be multiple certificates proving that the answer is indeed "yes"; in other words, there might be several accepting paths in the certifier.

  - The **UP** complexity class, a subclass of **NP**, specifies that any deterministic Turing machine used to certify whether an instance I has the answer "yes" admits a single certificate C proving that the answer is "yes"—in other words, a single accepting path.
  - A problem is designated as **NP-Complete** if it belongs to **NP**, and any other problem in **NP** can be reduced to it in polynomial time (via a Cook reduction).

- Similarly, the **co-NP** complexity class includes problems for which any "no" instance can be certified by a deterministic Turing machine in polynomial time.

  - The **co-UP** class mirrors the **UP** property but for "no" instances.

- The **PSPACE** complexity class encompasses problems solvable by a deterministic Turing machine using a polynomial amount of space. Notably, **NP**$\subset$**PSPACE**. However, the conjecture remains that **NP**$\neq$**PSPACE**.

- A problem is deemed **PSPACE-Complete** if every other problem in **PSPACE** can be reduced to it through a Karp reduction (a more stringent version of a Cook reduction where the output of both problems must be the same).