COLLEGE ED MATHEMATIQUES
DOCTORAL STIC
PAYS DE LA LOIRE

Nantes Université

THÈSE DE DOCTORAT

NANTES UNIVERSITÉ

ÉCOLE DOCTORALE N° 641
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Mathieu BOLTEAU

Logic programs to infer computational models of human embryonic development

Thèse présentée et soutenue à Nantes, le 04 octobre 2024
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes (LS2N) - UMR 6004

Rapporteur et rapportrice avant soutenance :

Loïc PAULEVÉ — Directeur de recherche, CNRS, Bordeaux, France
Delphine ROPERS — Directrice de recherche, Inria, Grenoble, France

Composition du Jury :

Président :	Richard REDON	Directeur de recherche, Inserm, Nantes, France
Examinateurs et Examinatrices :	Clémence FRIOUX	Chargée de recherche, Inria, Bordeaux, France
	Cédric LHOUSSAINE	Professeur, Université de Lille, Lille, France
	Loïc PAULEVÉ	Directeur de recherche, CNRS, Bordeaux, France
	Richard REDON	Directeur de recherche, Inserm, Nantes, France
	Delphine ROPERS	Directrice de recherche, Inria, Grenoble, France
Dir. de thèse :	Jérémie BOURDON	Professeur, Nantes Université, Nantes, France
Co-enc. de thèse :	Carito GUZIOLOWSKI	Maîtresse de conférence, École Centrale de Nantes, Nantes, France

Invité :

Laurent DAVID — Maître de conférence-Praticien hospitalier, Nantes Université, Nantes, France

"Be who you are and say what you feel, because those who mind don't matter and those who matter don't mind."

— Bernard M. Baruch (1870-1965)

Remerciements

Acknowledgements

C'est tout naturellement que je souhaite adresser mes premiers remerciements à mes encadrent·e·s de thèse. Mes sincères remerciements à Jérémie Bourdon, mon directeur de thèse. Ton savoir, ton expérience et ton recul ont été pour moi très enrichissants durant ces trois dernières années. Mes chaleureux remerciements à Carito Guziolowski, mon encadrante de thèse. Merci pour ta bienveillance, ton expertise et ton investissement durant ma thèse. J'ai grandement apprécié travailler à tes côtés. À vous deux, merci pour votre soutien indéfectible, que ce soit dans les hauts comme dans les bas. Merci également à Laurent David pour ton encadrement, ta collaboration et ton expertise biologique tout au long de cette thèse. Ta passion débordante pour la science est pour moi un véritable modèle et une grande source d'inspiration.

Je remercie Loïc Paulevé et Delphine Roppers d'avoir accepté d'être rapporteur et rapportrice de mon manuscript de thèse. Merci également à Clémence Frioux, Cédric Lhoussaine et Richard Redon d'avoir accepté de participer à mon jury de soutenance.

Je remercie également les membres de mon CSI, Clémence Frioux et Geneviève Dupont, pour nos échanges très constructifs et le temps que vous m'avez consacré.

Je souhaite remercier l'équipe ComBi (Computational Biology) du LS2N qui m'a accueilli durant ces trois dernières années. Un remerciement particulier pour Damien, Géraldine, Samuel et Guillaume pour nos nombreux échanges. J'ai beaucoup appris sur le métier de chercheur et d'enseigant-chercheur grâce à vous.

Merci Alban pour ton aide avec pyBRAvo et l'installation en local de Pathway Commons.

Merci au LS2N et aux équipes administratives et techniques sans qui je n'aurait pas pu réaliser ma thèse. Des remerciements particuliers pour Alexiane, Virginie D., Charlery et Haritiana.

Merci aux étudiants et étudiantes de l'École Centrale de Nantes que j'ai pu encadrer lors de leurs projets. Nos collaborations et le travail que vous avez réalisé ont été très enrichissants pour ma thèse et moi-même. Merci également à mes étudiants et étudiantes, à qui j'ai pu donner cours, de m'avoir fait aimer l'enseignement.

Comment une thèse pourrait-elle bien se dérouler sans des collègues extraordinaires devenu·e·s des ami·e·s ? À mes yeux, c'est impossible ! La thèse est une étape de la vie qui met à rude épreuve la santé des doctorant·e·s, à la fois, physiquement (*mon genou en est témoin*), mais surtout mentalement. Un bon entourage et un fort soutien m'ont été d'une aide précieuse tout au long de ces dernières années. Pour cela, je me dois de réaliser des remerciements personnels, mais l'exhaustivité n'est pas garantie.

Merci à Josselin avec qui j'ai partagé mes trois ans de thèse. Nos longues discussions, nos parties de Yahtzee, Hanabi et Love Letter, ainsi que les bières au Berlin ont été de très bons moments pour moi. Un merci à mes co-bureaux de début thèse, le duo (inséparable), Anna et Marinna. Vous avez été un véritable moteur au cours de cette thèse. Votre bonne humeur et vos rires (parfois bruyants, *coucou Marinna*) ensoilleyaient mes journées. Note à moi même: ne pas ramener de bonbons avec vous. Merci à toi Mathieu. Tu m'as fait découvrir l'escalade que j'aime beaucoup désormais. Je ne comprenais pas toujours les jeux vidéos auxquels tu jouaies mais tu as toujours fait l'effort de me les expliquer, merci. Merci à toi Rémi, le "Papa" du groupe, pour nos (très) longues discussions, mais à la fois *si* passionnantes. Julie, merci pour ta bonne humeur et ta joie de vivre. Garde ton investissement dans ce que tu entreprends, c'est une qualité précieuse. Jolan, bien que l'on ne se soit pas vu autant que l'on aurait souhaité, merci d'avoir été là durant ces trois années. Sophie, tu as ta place dans mes remerciements. Tu as été une "grande soeur" de thèse, *un peu folle je dois l'avouer*, mais tu m'as beaucoup apporté en début de thèse, pour cela, merci. Merci à toi David pour nos échanges sur tant de sujets. Je te souhaite tout le meilleur pour le reste de ta thèse. Louis, tu es arrivé au cours de ma troisième année. Merci pour nos discussions au RU ou en cafet', ainsi que nos parties de jeux journalières le midi. Ali, merci pour ta bonne humeur (*et tes nombreuses anecdotes footballistiques*) lors de tes passages à la FST. Adrien, Antoine, Nico, Loriane, merci à vous. Ça a toujours été un plaisir d'aller boire un verre avec vous ! Merci à toi Thibault, je suis ravi d'avoir croisé ton chemin et te souhaite tout le meilleur pour la suite. Enfin, *un peu en vrac,* merci à Hind, Aurélien, Émile, Ben, Albane, Yasmina, et toutes les personnes que j'oublie.

Je souhaite remercier l'association LOGIN, dont j'ai eu l'honneur d'être le président pendant 2 ans. Cette association a été un formidable tremplin pour rencontrer des collègues au labo, en particulier des doctorants et doctorantes. J'ai consacré beaucoup de temps et d'énergie pendant ces 2 années, mais je suis fier de ce que nous avons accompli, avec l'aide et le soutien indéfectible de Josselin, Rémi, Julie, Thibault, David, Nico et Anna. J'espère que cette association perdurera encore longtemps. Je suis convaincu de son pouvoir et sa force auprès des doctorant·e·s. Merci à vous Malo, Julien et Xavier d'avoir repris le flambeau.

Un grand merci à la LD team. Les réunions (*dont je tairais le nom*) m'ont permis d'approfondir mes connaissances en biologie sur des sujets super intéressants. Discuter de bio avec vous à toujours été un réel plaisir, même si parfois vous avez réussi à me perdre. Merci à Éva, Océane, Constance, Émilie, Simon et tant d'autres.

Comment ne pas remercier mes copains et copines de longue date ? Malgré nos occasions de nous voir peu fréquentes parfois, vous avez été un véritable échappatoire au cours de ces dernières années. Provenant d'horizons totalement differents, vous êtes des personnes superbes. Merci d'avoir été là, même si vous ne compreniez pas toujours ce que je faisais. Merci à Élie, Sarah, Justine, Tiphaine, Louise, Ben, Thomas, Kévin, Élise, Ronan, Marina, Victor, Romane.

Je tiens à exprimer un immense merci à ma famille qui a été présente tout au long de cette thèse. Merci Maman, merci Papa, merci Nathan, et plus largement merci à toute ma famille, pour votre soutien constant, même si, vous aussi, vous ne compreniez pas toujours ce que je faisais. Je vous suis très reconnaissant de votre compréhension face à ma faible disponibilité auprès de vous ces derniers temps.

Enfin, un grand merci à toutes les personnes que j'ai pu rencontrer et qui m'ont beaucoup apporté tout au long de cette thèse, ainsi que par le passé. Je pense notamment à François, Tom, Marine, Louis, Valentine, Alex, Martin, Vivian, Baptiste, Thomas. Merci à celles et ceux que j'aurais pu oublier dans ces remerciements, j'espère que vous ne m'en tiendrez pas rigueur.

RÉSUMÉ EN FRANÇAIS

FRENCH ABSTRACT

Introduction

Contexte

En 2015, Lewis Wolpert, Cheryll Tickle et Alfonso Martinez Arias ont déclaré : "*Comprendre comment les embryons se développent est un énorme défi intellectuel, et l'un des objectifs ultimes de la science de la biologie du développement est de comprendre comment nous, les humains, nous nous développons.*" [1] [1]. Plusieurs facteurs expliquent la nécessité de mieux comprendre le développement embryonnaire. Certains mécanismes de régulation restent encore à comprendre et l'émergence de stades de développement spécifiques doit faire l'objet d'analyses plus détaillées. En outre, une compréhension plus approfondie permettra d'améliorer les techniques de procréation assistée, telles que la fécondation in vitro (FIV), en optimisant les conditions de culture des embryons. Lorsque les parents rencontrent des difficultés pour concevoir un enfant, ils peuvent faire appel à la FIV, qui consiste à cultiver des embryons in vitro (dans des boîtes en plastique en laboratoire) pendant 5 à 6 jours avant de transférer l'embryon dans l'utérus de la femme. Ce processus nécessite un milieu de culture spécifique et diverses méthodes d'évaluation sont utilisées pour sélectionner les embryons à implanter. Bien qu'elle soit pratiquée depuis plusieurs décennies, le taux de réussite est relativement faible, de l'ordre de 20 à 30% [2]. Il est donc crucial d'améliorer notre compréhension du développement embryonnaire humain.

L'étude de l'embryon humain est complexe, en raison de différents facteurs.

Tout d'abord, le cadre juridique est très strict, et la culture d'embryons doit respecter de nombreuses réglementations. En France, les activités de recherche sur les embryons sont encadrées et contrôlées par l'Agence de la biomédecine. La recherche doit utiliser

1. Traduction de "*Understanding how embryos develop is a huge intellectual challenge, and one of the ultimate aims of the science of developmental biology is to understand how we humans develop.*" [1]

des embryons issus de FIV et donnés à la recherche, avec des restrictions telles que l'interdiction de cultiver les embryons pendant plus de 14 jours après la fécondation ou d'introduire des cellules d'une autre espèce dans un embryon humain.

Deuxièmement, la culture d'embryons humains in vitro requiert des compétences techniques hautement spécialisées, notamment en ce qui concerne l'élaboration de protocoles de culture et la manipulation des embryons. De plus, la composition des milieux de culture joue un rôle important dans le développement de l'embryon.

Troisièmement, les embryons humains sont biologiquement complexes. Les mécanismes de régulation qui interviennent au cours du développement sont spécifiques et suivent une orchestration qui n'est pas encore totalement comprise. Le nombre limité d'embryons et la variabilité des échantillons compliquent également les analyses.

Ces défis soulignent la nécessité de trouver des solutions alternatives pour étudier le développement de l'embryon humain. Ces solutions ne visent pas à remplacer les analyses biologiques des embryons humains, mais à fournir des informations complémentaires. Les deux approches essentielles pour l'étude du développement embryonnaire humain sont : *(i)* les modèles cellulaires et *(ii)* les modèles informatiques.

Pour modéliser biologiquement les embryons humains, les cellules souches peuvent être utilisées pour représenter des stades de développement ou des destins cellulaires spécifiques [15, 16]. Ces cellules ont la capacité de se différencier en plusieurs types cellulaires et peuvent être génétiquement modifiées pour imiter des stades ou des lignées embryonnaires spécifiques. En 2022, une lignée cellulaire spécifique issue de cellules souches a été utilisée pour former des blastoïdes. Un blastoïde est une structure cellulaire composée de trois lignées cellulaires (trophectoderme, épiblaste et endoderme, voir la section suivante pour plus de détails) formant un blastocyste, un stade de développement de l'embryon. Le blastoïde est morphologiquement similaire à un embryon et est capable de s'implanter dans l'utérus [3]. Ces blastoïdes, qui peuvent être générés en grand nombre, constituent une ressource prometteuse pour pallier le manque d'embryons disponibles et tester des hypothèses avec moins de contraintes juridiques.

Une autre approche pour modéliser le développement embryonnaire humain est la bioinformatique, qui utilise la biologie des systèmes pour déduire des modèles informatiques. Ces modèles permettent aux chercheurs et chercheuses de changer leur point de vue sur le système modélisé, ce qui peut leur permettre de découvrir de nouvelles connaissances sur le développement embryonnaire humain. En outre, les modèles

informatiques offrent des capacités inestimables pour prédire l'impact des perturbations sur ces systèmes. L'utilisation d'approches informatiques permet également de surmonter les limites biologiques et est très utile pour effectuer des simulations basées sur des hypothèses.

Cette thèse s'inscrit dans un projet de recherche qui vise à déduire des modèles informatiques du développement embryonnaire humain. Nous nous concentrons sur le développement préimplantatoire pour deux raisons. Premièrement, il s'agit de la partie la mieux comprise du développement embryonnaire précoce, ce qui la rend appropriée pour valider les résultats de notre modélisation. Deuxièmement, c'est la partie qui est reproduite in vitro lors de la FIV. Par conséquent, la modélisation de cette étape sera bénéfique pour les améliorations futures de la FIV.

Dans les sections suivantes, nous présentons d'abord le développement préimplantatoire de l'embryon humain, puis nous exposons l'objectif de cette thèse.

Développement préimplantatoire de l'embryon humain

Le développement embryonnaire humain commence par la fécondation d'un ovocyte par un spermatozoïde (Figure 1A). Le zyogte fécondé se divise ensuite toutes les 24 heures environ, formant la morula au quatrième jour. À ce stade, les cellules commencent à se différencier en divers types. Le développement préimplantatoire se poursuit jusqu'à l'implantation de l'embryon dans l'utérus de la femme. Au cours de ce développement, plusieurs événements clés se produisent.

Le premier événement majeur est l'activation du génome zygotique (ZGA) qui se produit au stade 8 cellules (*8 cells*). Il est suivi par la première spécification cellulaire, où la masse cellulaire interne (ICM) se sépare du trophectoderme (TE), marquant la décision initiale du destin des cellules. Une deuxième spécification au sein de l'ICM conduit à la formation de l'épiblaste (EPI) et de l'endoderme primitif (PrE), créant ainsi deux destins cellulaires distincts. Finalement, ces trois destins cellulaires, TE, EPI et PrE, sont présents lors de la phase d'implantation.

Chacun de ces destins cellulaires a un rôle spécifique dans le développement. Le TE formera le placenta, le PrE deviendra le sac vitellin du fœtus et l'EPI est destinée à être le futur fœtus. Les étapes du développement embryonnaire préimplantatoire peuvent être résumées dans le schéma présenté dans la Figure 1B.

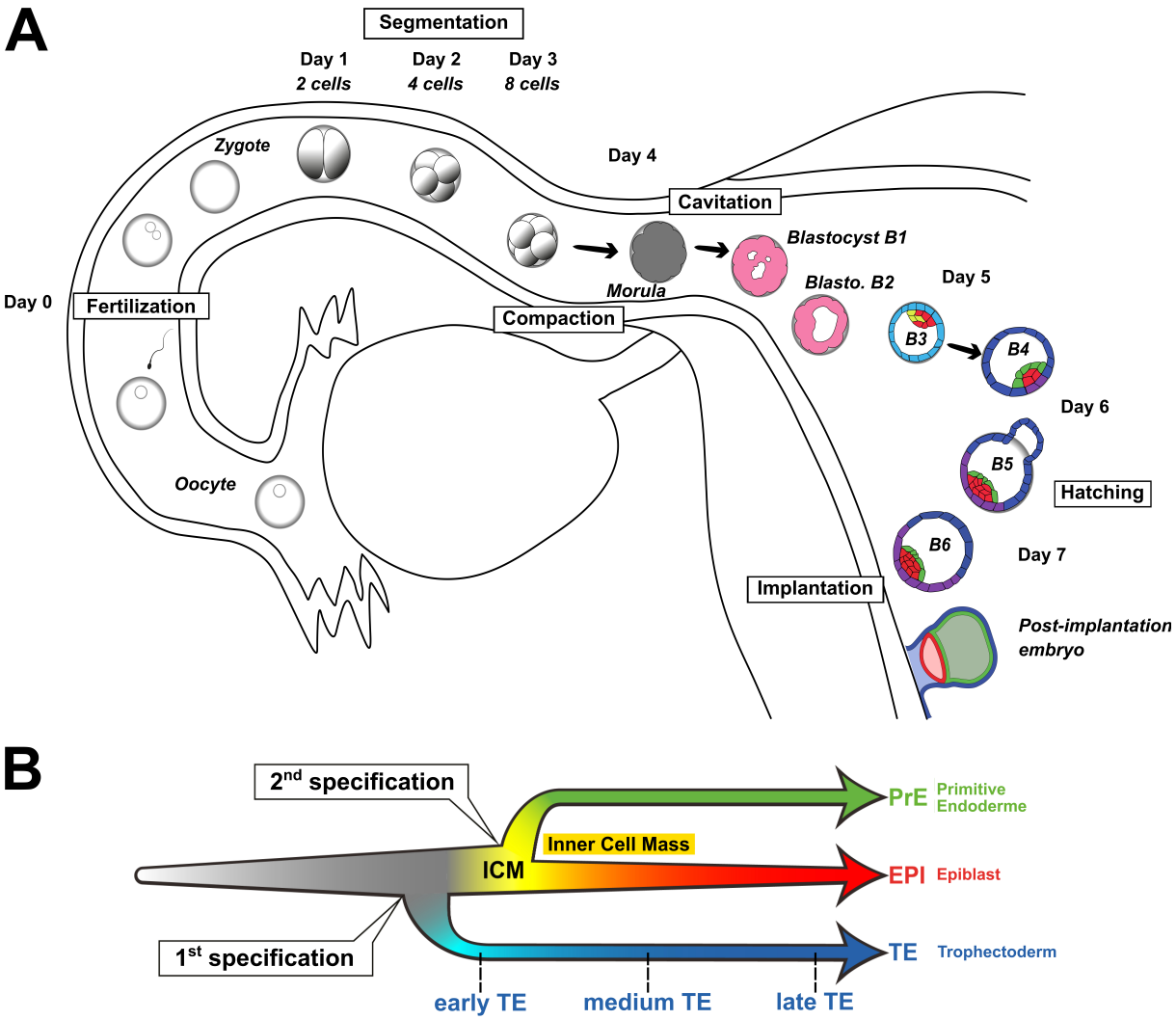


Figure 1 – Le développement embryonnaire préimplantatoire humain.
(A) Les différentes étapes du développement préimplantatoire. L'ovocyte est fécondé par un spermatozoïde, formant un zygote. Le zygote se divise environ toutes les 24 heures, se transformant en morula. Les cellules commencent à se différencier en suivant deux spécifications successives, ce qui donne lieu à trois destins cellulaires. L'embryon s'implante dans l'utérus de la femme. Notons que les couleurs des cellules correspondent aux destins cellulaires représentés en (B).
(B) Représentation schématique. Au cours du développement préimplantatoire de l'embryon humain, deux spécifications se produisent, conduisant à trois destins cellulaires : l'endoderme primitif (PrE), l'épiblaste (EPI) et le trophectoderme (TE). Dans la maturation du TE, trois phases sont observées : TE précoce, TE moyen et TE tardif.
Traduction: *day*, jour; *fertilization*, fécondation; *oocyte*, ovocyte; *cell*, cellule; *blastocyst*, blastocyste; *hatching*, éclosion, *Post-implantation embryo*, embryon post-implanté; *specification*, spécification; *early*, précosse, *medium*, moyen; *late*, tardif; *inner cell mass*, masse cellulaire interne; *primitive endoderm*, endoderme primitif; *epiblast*, épiblate; *trophectoderm*, trophectoderme.
Figure adaptée de Meistermann [4].

Objectif

Cette thèse s'inscrit dans un projet de recherche visant à définir les processus dynamiques impliqués dans la différenciation cellulaire, conduisant aux trois destins cellulaires : épiblaste, endoderme primitif et trophectoderme. En analysant les mécanismes de régulation impliqués dans le développement embryonnaire, une meilleure compréhension du développement pourrait permettre d'optimiser les conditions de culture embryonnaire et pouvoir ainsi augmenter les taux de réussite de la FIV.

Dans ce contexte, l'objectif de cette thèse est de comprendre les mécanismes de régulation des gènes impliqués dans le développement embryonnaire humain. En considérant plusieurs stades de développement, la question centrale est de savoir comment l'embryon passe d'un stade à l'autre et quels sont les mécanismes qui influencent ces décisions. Bien que l'on sache que certains gènes sont importants à certains stades du développement, les mécanismes de régulation entre ces gènes (et les autres) ne sont pas toujours clairs. Les mécanismes de régulation entre ces gènes (et d'autres potentiellement inconnus) restent donc à découvrir.

Cet objectif présente plusieurs défis. Le premier défi est lié aux données. Nous utilisons des données transcriptomiques de cellules uniques comprenant un grand nombre de cellules et de gènes à considérer (environ 1 500 cellules et plus de 20 000 gènes ; cf. Section 4.2.3.1). Ces données sont bruitées et contiennent une forte concentration de valeurs nulles, ce qui rend l'analyse complexe (cf. Section 2.2.3). Le deuxième défi concerne le système étudié, qui ne peut être perturbé. Cela complique la tâche de modélisation et de validation des résultats obtenus.

Dans cette thèse, nous nous concentrerons principalement sur la maturation du destin du trophectoderme (TE). Le TE passe par trois phases de maturation : précoce, moyenne et tardive (Figure 1B). En outre, le TE, en particulier dans sa phase tardive, est responsable de l'attachement de l'embryon à l'utérus, un rôle crucial pour le développement futur du fœtus. Par conséquent, une meilleure compréhension de cette maturation et le développement de modèles informatiques du TE fourniront des informations précieuses pour le domaine.

Vue d'ensemble de notre approche de modélisation

Dans cette section, nous présentons la vue d'ensemble de notre approche de modélisation. Nous considérons deux stades de développement, par exemple le stade TE

moyen et le stade TE tardif. Nous posons le postulat qu'une cellule TE moyen peut rester à ce stade ou se différencier en stade TE tardif (Figure 1.2A). La cellule TE moyen peut être considérée comme un état initial, tandis que la cellule TE tardif peut être considérée comme un état final. Considérons ce problème comme un problème de perturbation. Nous disposons d'un ensemble de perturbations effectuées sur le système et des observations capturées aux états initial et final. Étant donné toutes les transitions possibles, l'objectif est de trouver des chemins expliquant comment le système peut passer des perturbations aux observations. Les chemins, spécifique aux états, serviront à modéliser les états étudiés. Par conséquent, les deux modèles expliquent comment le système peut passer des perturbations aux observations.

Si l'on considère les données transcriptomiques unicellulaires, chaque cellule a une expression spécifique pour chaque gène (Figure 2B). L'idée est d'identifier les similitudes et les différences, en termes d'expression génique, entre les deux états (ou cellules). Les similitudes peuvent être considérées comme des pseudo-perturbations et les différences comme des pseudo-observations (Figure 2C). Ici, nous prenons l'exemple d'une cellule; cependant, plusieurs cellules forment des étapes, ce qui donne un ensemble de paires pseudo-perturbations et pseudo-observations qui doivent être prises en compte. Enfin, compte tenu de toutes les interactions géniques possibles combinées aux pseudo-perturbations et aux pseudo-observations, des chemins pseudo-perturbations–pseudo-observations sont appris, modélisant chaque état (Figure 2D).

Contributions

Au cours de cette thèse, une partie des travaux que j'ai menés a été publiée. Durant ces trois dernières années, j'ai eu l'occasion d'être le premier auteur de deux publications et de contribuer en tant que co-auteur à une troisième publication. De plus, une quatrième publication, dont je serai le premier auteur, est en cours de rédaction et sera soumise à un journal.

La première publication, Bolteau *et al.* [5], présentée dans le Chapitre 4, introduit la méthode que nous avons développée pour inférer des réseaux Booléens à partir de connaissances préalables et de données transcriptomiques de cellules uniques (scRNAseq). Ce travail a été publié dans les actes de conférence du 19e *International Symposium on Bioinformatics Research and Applications* (ISBRA 2023). J'ai également présenté nos travaux lors de la conférence qui s'est tenue à Wrocław, en Pologne. Notre objectif

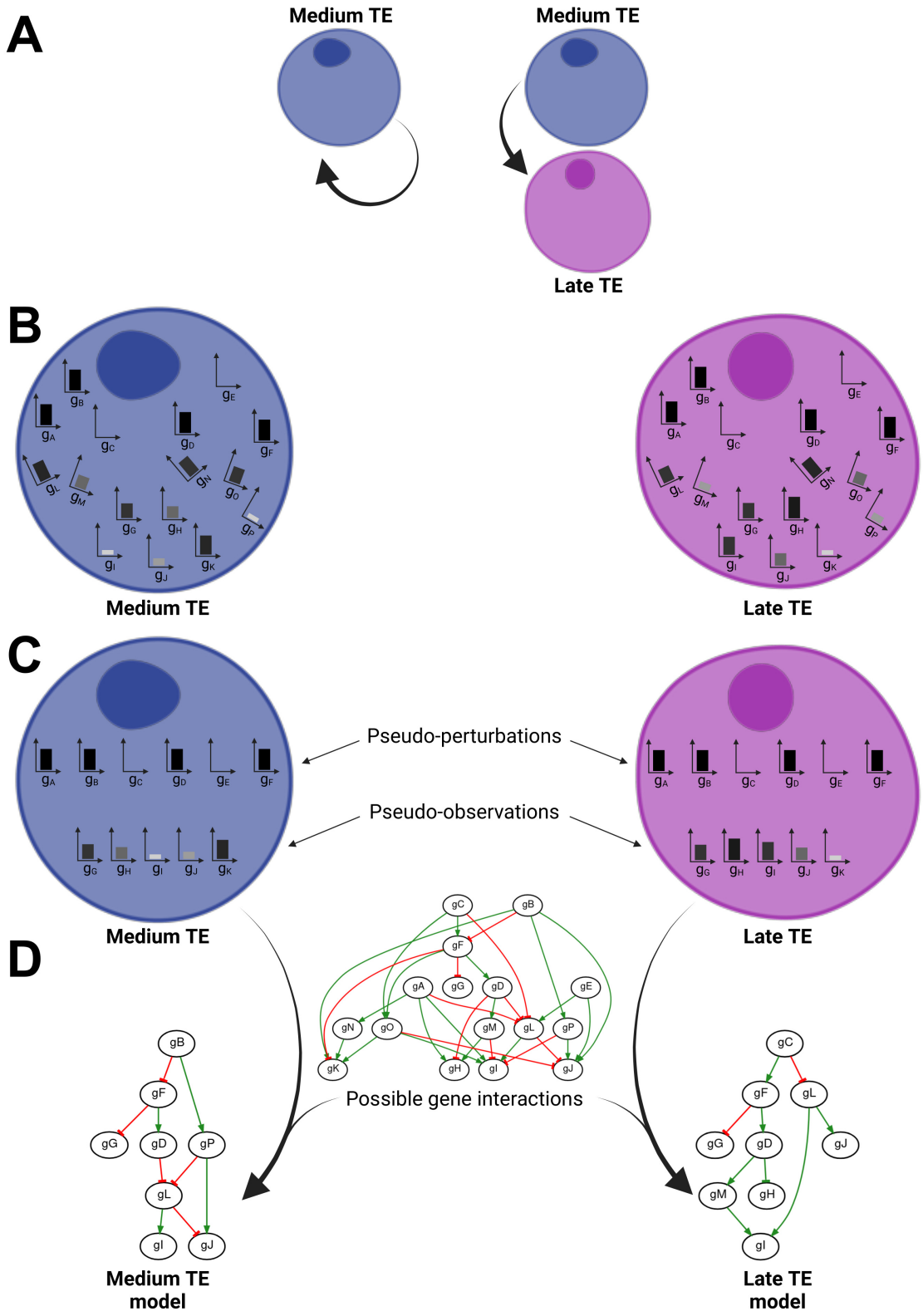


Figure 2 – Vue d'ensemble de notre approche de modélisation.
(légende sur la page suivante.)

dans cette contribution est d'explorer le développement embryonnaire humain, en nous concentrant spécifiquement sur la compréhension de la maturation du trophectoderme (TE). Nous utilisons des données scRNAseq pour développer un framework permettant d'inférer des modèles informatiques qui distinguent deux stades de développement impliqués dans le développement de l'embryon. Notre méthode sélectionne des pseudo-perturbations à partir des données scRNAseq car les perturbations réelles ne sont pas réalisables en raison de contraintes éthiques et légales. En combinant ces pseudo-perturbations avec des réseaux de régulation, nous pouvons déduire des réseaux Booléens qui s'alignent avec précision sur les données scRNAseq pour chaque stade de développement étudié. Notre méthode, accessible publiquement, a été testée à l'aide de plusieurs benchmarks, ce qui prouve la faisabilité de notre approche. Appliquée à l'ensemble des données réelles, nous déduisons des familles de réseaux Booléens correspondant aux stades de développement moyen et tardif du TE. Leurs structures révèlent des voies de régulation contrastées, offrant des perspectives et des hypothèses biologiques précieuses dans ce domaine.

La seconde publication, Bolteau *et al.* [6], présentée dans le Chapitre 4, a été publiée dans le *Journal of Computational Biology* en 2024. Cette contribution propose une exploration plus approfondie des résultats du premier article. Ici, nous utilisons des données scRNAseq pour modéliser les mécanismes de régulation des gènes impliqués dans deux stades de développement humain : le trophectoderme (TE) moyen et le TE tardif.

(suite de la page précédente.)
(A) Une cellule au stade TE moyen peut soit rester au stade TE moyen, soit se différencier au stade TE tardif.
(B) Chaque cellule présente un profil d'expression spécifique pour chaque gène. Certains gènes sont exprimés de manière similaire dans les deux cellules (e.g., g_A, g_B, g_C, g_D, g_E, g_F), tandis que d'autres présentent une expression différentielle.
(C) Les pseudo-perturbations sont identifiées (g_A, g_B, g_C, g_D, g_E, g_F). Certains autres gènes forment les pseudo-observations (g_G, g_H, g_I, g_J, g_K). Nous prenons ici l'exemple d'une cellule ; cependant, plusieurs cellules forment des stades, ce qui donne un ensemble de paires pseudo-perturbations et pseudo-observations qui doivent être prises en compte.
(D) Étant donné toutes les interactions géniques possibles dérivées de la connaissance préalable, combinées aux pseudo-perturbations et aux pseudo-observations extraites pour toutes les cellules individuelles à chaque stade, des modèles spécifiques au stade sont appris. Les interactions d'activation sont représentées par des flèches vertes classiques ($\longrightarrow$), tandis que les interactions d'inhibition sont représentées par des "flèches en T" rouges ($\longrightarrow\!\!\dashv$).
Traduction: *medium*, moyen; *late*, tardif; *possible gene interactions*, interactions de genes possibles; model, modèle.
Figure créée avec BioRender.com.

Nous identifions 20 pseudo-perturbations, qui sont intégrées aux interactions géniques déjà connues pour déduire des réseaux Booléens spécifiques à chaque stade. Ces réseaux Booléens délimitent des mécanismes de régulation distincts, ce qui permet de différencier ces stades de développement. Nous montrons que notre programme est plus performant que les outils d'identification des pseudo-perturbations existants. Notre méthode contribue à la compréhension des processus de développement humain et peut s'appliquer à divers stades de développement et à d'autres domaines de recherche.

La troisième publication, Le Bars *et al.* [7], a été publiée dans le journal *BMC Bioinformatics* en 2023. Dans cette publication, où je suis co-auteur, nous présentons MajS, une méthode visant à améliorer la modélisation des réseaux de régulation pour faciliter leur intégration avec les réseaux métaboliques. Étant donné un réseau de régulation et un ensemble partiel discret d'observations en entrée, MajS teste la cohérence entre les données d'entrée, propose des réparations minimales sur le réseau pour établir la cohérence, et enfin calcule des prédictions pondérées et signées sur les espèces du réseau. Nous avons testé MajS en comparant la voie de signalisation HIF-1 avec deux ensembles de données d'expression génique. Nos résultats montrent que MajS peut prédire 100% des espèces non observées. En comparant MajS avec deux outils similaires (discret et quantitatif), nous avons observé que par rapport à l'outil discret, MajS propose une meilleure couverture des espèces non observées, est plus sensible aux perturbations du système, et propose des prédictions plus proches des données réelles. Par rapport à l'outil quantitatif, MajS fournit des prédictions discrètes plus raffinées qui sont en accord avec la dynamique proposée par l'outil quantitatif. MajS est une nouvelle méthode permettant de tester la cohérence entre un réseau de régulation et un ensemble de données qui fournit des prédictions informatiques sur des espèces de réseau non observées. Elle fournit des prédictions discrètes à grain fin en donnant le poids du signe prédit en tant qu'élément d'information supplémentaire. Les résultats de MajS, grâce à leur poids, peuvent être facilement intégrés à la modélisation des réseaux métaboliques.

Enfin, une quatrième publication est en cours de rédaction et sera soumise à un journal de bioinformatique prochainement. Cet article présentera les résultats prêts à être publiés du Chapitre 5, mettant en évidence des réseaux Booléens plus robustes et plus pertinents modélisant le stade moyen et tardif du TE. Ce papier mettra en évidence notre méthode, SCIBORG, comprenant les améliorations par rapport à la méthode utilisée dans les deux premiers articles publiés.

Plan du manuscrit

Ce manuscrit est organisé autour de cinq chapitres.

Le Chapitre 2 traite du paysage actuel de la modélisation des données de cellules uniques. Il comprend une vue d'ensemble du séquençage transcriptomique de cellules uniques et une analyse de trois outils de modélisation.

Le Chapitre 3 présente les définitions nécessaires et donne un aperçu des données utilisées. Il passe également en revue le paradigme de programmation et les différentes méthodes utilisés dans cette thèse.

Le Chapitre 4 détaille la méthode mise en œuvre pour modéliser les étapes du développement à l'aide de l'inférence de réseaux Booléens. Ce chapitre met en évidence les principaux résultats présentés dans les deux articles publiés, à savoir: Bolteau *et al.* [5] et Bolteau *et al.* [6].

Le Chapitre 5 met en évidence les améliorations que nous avons apportées à la méthode précédente, ce qui a permis d'obtenir des résultats plus robustes. Cette deuxième contribution fera l'objet d'un prochain article qui sera soumis à un journal.

Le Chapitre 6 offre une discussion générale du travail présenté dans ce manuscrit et esquisse les perspectives de cette thèse.

Discussion et Perspectives

Discussion

Dans ce manuscrit, nous présentons deux contributions réalisées dans le cadre de cette thèse. La première introduit une méthode d'inférence de réseaux Booléens (*Boolean networks*, BNs) visant à modéliser deux stades du développement embryonnaire humain. Dans la seconde contribution, nous décrivons les améliorations qui conduisent à l'énumération exhaustive et aux familles optimales de modèles, à travers une méthode nommée SCIBORG. L'assemblage de ces deux contributions a conduit à SCIBORG, un outil original qui permet de mettre en évidence les principaux mécanismes impliqués dans le développement embryonnaire humain. SCIBORG se compose de trois étapes : *(i)* la reconstruction du réseau de connaissances préalables (PKN), *(ii)* la construction des plans d'expériences, et *(iii)* l'inférence des BNs. Tout d'abord, la reconstruction du PKN fait appel à un outil qui interroge la base de données Pathway Commons afin de reconstruire un graphe d'interactions géniques constituant nos connaissances préalables.

Deuxièmement, pour construire des plans d'expériences spécifiques à chaque stade, nous utilisons d'abord un programme *Answer Set Programming* (ASP) pour identifier les pseudo-perturbations. Ensuite, nous cherchons à maximiser les différences des readouts entre les cellules sélectionnées par les pseudo-perturbations. Troisièmement, étant donné le PKN et les plans expérimentaux, nous utilisons un outil codé en ASP pour déduire des BNs qui sont à la fois compatibles avec les interactions géniques présentes dans le PKN et l'expression génique présente dans les plans expérimentaux.

Appliqué aux stades moyen et tardif du TE, SCIBORG nous permet d'identifier un PKN comprenant 233 gènes et complexes protéiques, et 369 interactions géniques. Notre programme d'identification des pseudo-perturbation permet d'identifier 96 pseudo-perturbations dans les deux stades étudiés. En explorant les solutions équivalentes aux pseudo-perturbation, nous avons trouvé deux solutions composées de deux sous-ensembles de 10 gènes permettant l'identification de 96 pseudo-perturbations; c'est-à-dire 96 cellules dans chaque lignée dont le profil d'expression nous a permis de différencier les stades TE moyen et tardif. Nous considérons deux méthodologies de normalisation des readouts et identifions que la normalisation "arctangeante" permet l'apprentissage de résultats plus robustes et pertinents. Dans la troisième étape, nous avons appris deux familles de BNs modélisant le TE moyen et le TE tardif. Nous identifions différents mécanismes de régulation, spécifiques aux stades étudiés. De plus, nos BNs inférées permettent l'introduction d'un classificateur cellulaire, visant à classer une cellule dans le stade le plus proche, en fonction de son expression génique.

Enfin, les résultats de la deuxième contribution, présentés dans le Chapitre 5, seront compilés dans un article scientifique et soumis à un journal de bioinformatique pour publication.

Pré-traitement des données

Dans l'étape de pré-traitement, les données sont soit binarisées, pour les gènes d'entrée et les gènes intermédiaires, soit normalisées pour les gènes readouts.

Nous utilisons une méthode de binarisation simple, considérant qu'un gène est exprimé si au moins deux *reads* sont observées dans la cellule. Toutefois, cette approche est sujette à discussion. Il serait intéressant d'explorer différentes valeurs de seuil pour voir si elles ont un impact sur les résultats de la binarisation. Par ailleurs, d'autres méthodes de binarisation existent, comme PROFILE [8] utilisée dans BoNesis, ou RefBool [9]. Ces méthodes peuvent cependant attribuer des niveaux d'expression intermédiaires à

certains gènes, ou écarter certains gènes. L'application de ces méthodes en l'état pourrait potentiellement entraîner l'exclusion de certains gènes ou cellules (lorsqu'un gène n'est pas pris en compte), ce qui ne peut pas être géré pour le moment dans SCIBORG.

En ce qui concerne la normalisation des readouts, la méthodologie utilisée a un impact significatif sur les BNs déduits (multiplication par deux de la précision en comparant la normalisation "arctangente" à la normalisation "min-max"). La normalisation "arctangente" fournit des BNs plus robustes et plus pertinents. Néanmoins, d'autres méthodes de normalisation pourraient être explorées afin d'obtenir de meilleurs résultats.

Reconstruction du PKN

Notre méthode de reconstruction du PKN s'appuie sur pyBRAvo qui interroge Pathway Commons, une ressource regroupant plusieurs bases de données. Pathway Commons inclut KEGG [10], la base de données utilisée dans l'étude de Chebouba *et al.* [11], ce qui nous permet d'incorporer plus de connaissances préalables que celles utilisées dans leur étude. Nous avons choisi d'utiliser une base de données externe pour reconstruire le PKN au lieu d'une méthode d'inférence de PKN utilisant des données scRNAseq, telle que LEAP (cf. Chapitre 2, Section 2.4), car le PKN déduit à partir de ces méthodes peut être biaisé par les données.

De plus, notre PKN reconstruit comprend de nombreuses interactions géniques dérivées des données d'analyse du cancer. Cependant, les mécanismes impliqués dans le cancer diffèrent de ceux du développement embryonnaire, un problème qui doit être résolu. Une solution potentielle pourrait être l'utilisation de la base de données DoRothEA [12], telle qu'elle est employée dans BoNesis. Cette base de données contient des interactions avec des niveaux de confiance variables, allant d'un niveau de confiance élevé (interactions étudiées) à un niveau de confiance faible (prédictions), ce qui nécessite une utilisation prudente. Une autre option consisterait à utiliser une méthode d'inférence basée sur les données scRNAseq, en gardant toujours à l'esprit le biais potentiel des données dans le PKN.

Enfin, nous avons recosntruit deux PKN différents (PKN^0 et PKN^2), dont la taille varie. Il a été observé que le PKN le plus grand, PKN^0, produisait des liens biologiques inférés plus pertinents (voir Chapitre 5, Section 5.3.4). Malgré l'augmentation de la taille, SCIBORG peut traiter ce problème efficacement.

Utilisation d'autres PKNs

Il est important de noter que l'étape de reconstruction du PKN est facultative dans SCIBORG. Les utilisateurs et utilisatrices possédant un PKN adapté à leur étude de cas spécifique ont la possibilité de l'utiliser et d'exécuter les étapes suivantes de la méthode. Notre approche vise à être aussi adaptable que possible pour prendre en compte les différents PKNs spécifiques à l'utilisateur ou l'utilisatrice.

Amélioration de l'étape de maximisation des readouts

La maximisation de la différence des readouts est implémentée en Python et intervient après l'identification des pseudo-perturbations en ASP. L'algorithme est simple, il calcule les différences par paire dans les cellules redondantes (cf. Chapitre 4, Section 4.2.5.2). Bien que l'on puisse envisager d'améliorer ce processus avec d'autres algorithmes, ce n'est pas une priorité puisque le temps d'exécution est raisonnablement court (moins de 5 minutes sur un ordinateur portable).

Une autre option consisterait à incorporer des contraintes supplémentaires dans le programme ASP d'identification des pseudo-perturbations, afin de maximiser les différences de readouts directement dans le processus de résolution ASP. Toutefois, cette approche serait très gourmande en ressources, ce qui compliquerait et allongerait considérablement la recherche de pseudo-perturbations.

En outre, nous utilisons une maximisation des différences de readout pour traiter les redondances cellulaires, dans le but d'obtenir la plus grande différence entre les deux stades étudiés et, potentiellement, de mieux les distinguer. Cependant, d'autres critères pourraient être envisagés. Nous avons exploré un critère de valeur de "readout moyen", qui implique le calcul des valeurs de readouts moyennes de toutes les cellules redondantes. Les BNs appris avec ce critère n'étaient pas significativement différents de ceux obtenus avec l'approche de maximisation des readouts, ce qui nous a conduit à retenir cette dernière dans SCIBORG.

Paramètres de Caspo

Pour déduire les réseaux Booléens à l'aide de Caspo, nous avons utilisé différents paramètres.

Le premier paramètre considéré est la *fitness_tolerance*, qui vise à fournir une tolérance en termes de MSE pour l'apprentissage des réseaux Booléens. Dans nos deux contributions,

nous fixons la valeur de *fitness_tolerance* à $0,0001$, ce qui permet d'explorer au-delà du BN optimal jusqu'à une distance de $0,01\%$ par rapport à la MSE optimale. Nous avons choisi cette valeur parce que c'est celle qui fournit en moyenne les BNs les plus pertinents.

Le deuxième paramètre est la tolérance de taille, qui fixe un niveau de tolérance pour l'exploration des BNs avec un nombre de nœuds allant de la taille optimale jusqu'à une tolérance maximale définie par l'utilisateur ou l'utilisatrice ajoutée au nombre optimal de nœuds. Nous avons choisi de ne pas fournir de tolérance en termes de taille car notre objectif est de trouver des modèles Booléens minimaux.

Le troisième paramètre est la longueur, qui limite le nombre d'interactions entrantes dans une porte logique "ET". Nous avons fixé cette longueur à 2 afin de simplifier les mécanismes de régulation déduits, réduisant ainsi la complexité du problème.

L'exploration plus approfondie de ces paramètres en testant différentes valeurs pourrait potentiellement produire des résultats plus robustes.

Améliorer la composition des BNs inférés

La composition des interactions géniques déduites est parfois insatisfaisante. Dans notre première contribution, les BNs appris comprenaient des genes intermédiaires directement liés aux readouts (cf. Chapitre 4, Section 4.3.3), ce qui n'est pas biologiquement informatif. Pour résoudre ce problème, nous avons incorporé des contraintes supplémentaires dans le programme ASP d'identification des pseudo-perturbations de SCIBORG, afin de sélectionner les genes d'entrée et les gènes intermédiaires qui sont connectés (cf. Chapitre 5, Section 5.2.3, *Contrainte 4*). Cette approche a permis d'inférer des réseaux biologiques plus pertinents.

Cependant, les BNs appris restent déconnectés lors de l'utilisation de la normalisation "arctangeante" (cf. Chapitre 5, Section 5.3.4). Pour déduire des réseaux plus connectés, qui incluent des "cascades" des gènes d'entrée au readouts en passant par des gènes intermédiaires, le programme ASP de Caspo doit être modifié en profondeur. Cette tâche est difficile puisqu'elle nécessite une compréhension approfondie de chaque règle ASP pour apporter les modifications nécessaires, mais elle peut conduire à des résultats plus significatifs, ce qui est inestimable pour la modélisation.

Méthodes de l'état de l'art

Les méthodes de modélisation utilisant SMT ou ASP énumèrent une quantité massive de modèles possibles. Les méthodes de l'état de l'art traitent cette explosion de solutions en échantillonnant des modèles (BoNesis ou RE:IN) ou en s'appuyant sur des expériences. En revanche, SCIBORG permet d'identifier des modèles équivalents au cours du temps (cf. Chapitre 5, Section 5.3.3.1). Nous démontrons que le nombre de modèles possibles est très limité. Par rapport aux méthodes de l'état de l'art, notre espace de solution est plus restreint, avec seulement 2 modèles contre des centaines voire des milliers. Ces résultats fournissent un niveau élevé de confiance dans nos résultats et soutiennent fortement la validation expérimentale dirigée de nos modèles.

De plus, nous observons que SCIBORG prend en compte l'hétérogénéité des cellules par l'identification de plusieurs cellules (environ 100) au sein de chaque classe. Pour chaque cellule, nous considérons un panel d'environ 30 gènes, dont le profil d'expression nous permet d'apprendre les BNs. En revanche, BoNesis travaille avec une moyenne de cellules au sein d'une classe, tandis que RE:IN utilise des données en vrac. SCNS prend en compte les données de cellules uniques sur une fenêtre d'environ 40 gènes ; cependant, il ne gère pas l'expression redondante entre les cellules. La prise en compte de l'hétérogénéité cellulaire permet à SCIBORG de prendre en compte les redondances couramment présentes dans les données transcriptomiques de cellules uniques, un facteur qui n'est pas pris en compte dans les autres méthodes de l'état de l'art.

Perspectives

Les travaux menés au cours de cette thèse constituent une avancée significative dans la modélisation du développement embryonnaire humain. Bien que notre recherche se soit concentrée sur la modélisation du trophectoderme moyen et tardif (TE), il reste de nombreux domaines à explorer.

Tout d'abord, nous prévoyons d'analyser d'autres stades de développement impliqués dans le développement préimplantatoire humain. L'objectif premier est de poursuivre l'exploration de la maturation du TE en incluant le TE précoce, une phase qui se produit avant les autres. Par la suite, nous souhaitons étendre notre analyse à d'autres stades de développement afin d'étudier les autres destins cellulaires : l'endoderme primitif (PrE) et l'épiblaste (EPI).

Deuxièmement, nous prévoyons d'approfondir l'étude du classificateur cellulaire

proposé avec SCIBORG pour classer les cellules dans les stades correspondants. Cet outil pourrait aider à classer les cellules non définies qui apparaissent dans les analyses menées par Meistermann *et al.* [13] (cf. Chapitre 3, Section 3.2). Pour y parvenir, nous devons explorer les stades voisins de ces cellules, en nous concentrant en particulier sur l'EPI et le PrE, ce qui rejoint le point précédent.

Les BN déduits pourraient être challengés par simulation, par exemple en désactivant un gène et en observant le comportement des BNs et de l'expression génique qui en résulte. Cette approche puissante faciliterait les prédictions in silico, en fournissant des informations précieuses et en guidant éventuellement la validation expérimentale.

Parallèlement aux modèles informatiques que nous déduisons, les modèles biologiques, en particulier les blastoïdes (cf. Chapitre 1), sont des outils prometteurs pour la validation des modèles informatiques avec moins de contraintes légales. Les blastoïdes pourraient être très utiles pour valider in vitro les simulations in silico et confirmer ainsi les hypothèses posées.

Également, nous prévoyons d'appliquer SCIBORG à d'autres études biologiques. Nous collaborons avec une équipe du laboratoire CRCI^{2}NA de Nantes, qui travaille sur le développement des cellules lymphoïdes internes. Leur objectif est de comprendre les mécanismes de régulation des gènes impliqués dans la différenciation de ces cellules. SCIBORG pourrait contribuer à l'identification de mécanismes inconnus dans ce processus.

En fin de compte, SCIBORG facilite la modélisation statique par le biais des réseaux d'interactions déduits. La modélisation des processus dynamiques survenant au cours du développement embryonnaire constitue une orientation intéressante pour les travaux futurs. À cette fin, une extension de Caspo, connue sous le nom de Caspo Time Series (Caspo-ts) [14], pourrait être employée. Étant donné un réseau de connaissances préalables (PKN) et des données de séries temporelles, qui pourraient être déduites de l'expression génique pseudo-temporelle dans notre cas, Caspo-ts infère des BNs compatibles à la fois avec les interactions géniques définies dans le PKN et avec les modèles d'expression génique identifiés dans les données de séries temporelles.

Des travaux préliminaires ont été menés par des étudiants et étudiantes que j'ai co-supervises avec Carito Guziolowski, explorant cette approche et aboutissant à des résultats prometteurs. Cette perspective constitue la base d'une nouvelle thèse de doctorat qui

débutera prochainement. Je suis très heureux de constater que les travaux menés pendant ma thèse seront poursuivis et étendus.

Table of Contents

List of Figures

List of Tables

List of Programs

Chapter 1

Introduction

"The scientist is not a person who gives the right answers, he's one who asks the right questions."

— Claude Lévi-Strauss (1908-2009)

Summary

This chapter offers a global introduction of the manuscript. We provide the context of the thesis and then outline the human embryonic preimplantation development process. Then, we present the objective of the thesis and the overview of our modeling method. This chapter finishes with a presentation of the contributions made during the thesis and an outline of the manuscript.

1.1 Context

In 2015, Lewis Wolpert, Cheryll Tickle and Alfonso Martinez Arias stated, "*Understanding how embryos develop is a huge intellectual challenge, and one of the ultimate aims of the science of developmental biology is to understand how we humans develop.*" [1]. Several factors drive the need to better understand embryonic development. Certain regulatory mechanisms remain elusive, and the emergence of

specific developmental stages needs further detailed analyses. Additionally, a deeper understanding will help to enhance assisted reproductive technologies (ART), such as in vitro fertilization (IVF), by optimizing embryo culture conditions. When parents face difficulties conceiving, they can opt for IVF, which involves culturing embryos in vitro (in plastic boxes in a wet lab) for 5-6 days before transferring the embryo into the female uterus. This process requires a specific culture medium, and various evaluation methods are used to select the embryos for implantation. Despite being practiced for several decades, success rate of IVF is relatively low, around $20-30\%$ [2]. Thus, improving our understanding of human embryo development is crucial.

Study human embryo is complex, due to various factors.

First, the legal framework is very strict, and embryo culture must adhere to many regulation rules. In France, research activities on embryos are constrained and scrutinized by the Agence de la Biomédecine. Research must use embryos donated to the research from IVF, with restrictions such as not culturing embryos for more than 14 days post-fertilization or introducing cells from another species into a human embryo.

Second, cultivating human embryo in vitro requires highly specialized technical skills, including the development of culture protocols and the handling of embryos. In addition, the composition of culture media plays an important role in embryo development.

Third, human embryos are biologically complex. The regulatory mechanisms occurring during the development are specific and follow an orchestration that is not yet fully understood. The limited number of embryos and sample variability further complicate the analyses.

These challenges highlight the need for alternative solutions to study human embryo development. These solutions are not intended to replace the biological analysis of human embryos but to provide complementary insights. Two essential approaches for studying of human embryonic development are: *(i)* cellular models and *(ii)* computational models.

To biologically model human embryos, stem cells can be used to represent specific developmental stages or cell fates [15, 16]. These cells have the ability to differentiate into multiple cell types and can be genetically modified to mimic specific embryonic stages or lineages. In 2022, a specific cellular lineage from stem cells was used to form blastoids [3]. A blastoid is a cell structure composed of three cell lineages (trophectoderm, epiblast and endoderm; see next section for further details) forming a blastocyst, a developmental stage of the embryo. The blastoid is morphologically similar to an embryo and capable

to implant in the uterus [3]. These blastoids, which can be generated in a large number of copies, offer a promising resource to address the lack of available embryos and test hypotheses with less legal constraints.

Another approach to model human embryonic development is through computational biology, using systems biology to infer computational models. These models enable researchers to shift their perspective on the modeled system, potentially uncovering new insights into human embryonic development. In addition, computational models offer invaluable capabilities in predicting how perturbations impact such systems. Using computational approaches also helps to overcome biological limitations and is very useful for testing of hypothesis-driven research.

This thesis situates within a research project which aims to infer computational models of the human embryonic development. We focus on preimplantation development for two reasons. First, it is the most well-understood part of early embryo development, making it suitable for validating our modeling findings. Second, it is the part that is reproduced in vitro during IVF. Therefore, modeling this stage will be beneficial for future IVF improvements.

In the following sections, we first present the human embryonic preimplantation development, and then outline the objective of this thesis.

1.2 Human preimplantation embryonic development

Human embryonic development begins with the fertilization of an oocyte by a spermatozoon (Figure 1.1A). The fertilized zygote then divides approximately every 24 hours, forming the morula by day 4. At this stage, cells start to differentiate into various types. Preimplantation development continues until the implantation of the embryo into the female uterus. During this development, several key events will take place.

The first major event is zygotic genome activation (ZGA) which happens at the 8-cell stage. This is followed by the first cell specification, where the inner cell mass (ICM) segregates from the trophectoderm (TE), marking the initial fate decision by the cells. A second specification within ICM leads to the formation of the epiblast (EPI) and the primitive endoderme (PrE), creating two distinct cell fates. Ultimately, these three cell fates, TE, EPI and PrE, are present at the implantation phase.

Each of these cell fates has a specific role in development. The TE will form the

placenta, the PrE will become the yolk sac of the fetus and the EPI is destined to be the future fetus. The stages of preimplantation embryonic development can be summarized in the schema presented in Figure 1.1B.

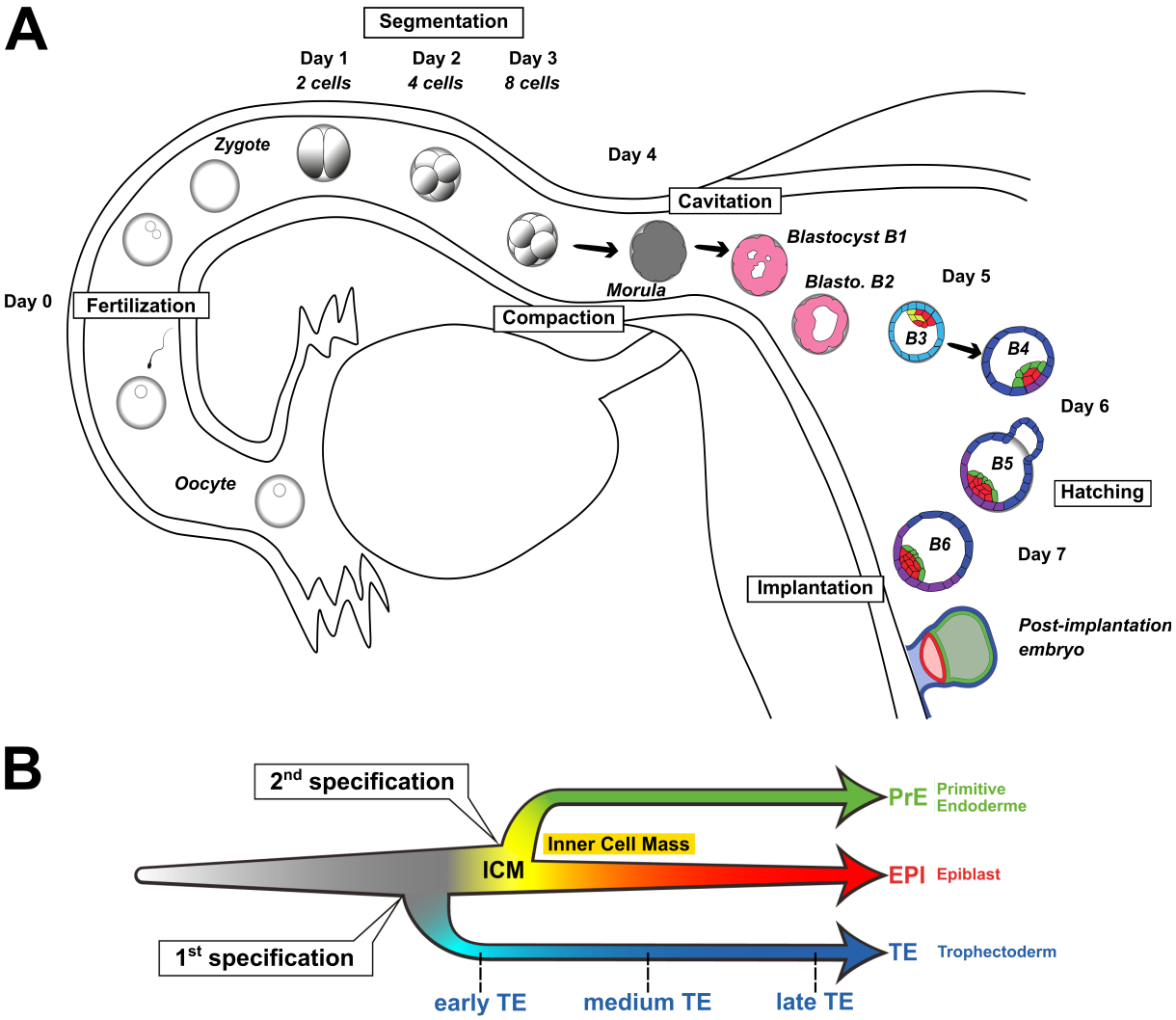


Figure 1.1 – The human preimplantation embryonic development.
(A) Different steps of the preimplantation development. The oocyte is fertilized by a spermatozoon, forming a zygote. The zygote divides approximately every 24 hours, developing into the morula. Cells start to differentiate through two successive specifications, resulting in three cell fates. The embryo implants into the female uterus. Notice that the colors of cells correspond to cell fates represented in (B). (B) Schematic representation. During the human embryonic preimplantation development, two specifications occur, leading to three cell fates: primitive endoderme (PrE), epiblast (EPI) and trophectoderm (TE). In the TE maturation, three phases are observed: early TE, medium TE and late TE.
Figure adapted from Meistermann [4].

1.3 Objective

This thesis is part of a research project aimed at defining the dynamic processes involved in cellular differentiation, leading to the three cell fates: epiblast, primitive endoderm, and trophectoderm. By analyzing the regulatory mechanisms involved in embryonic development, a better understanding of development could lead to optimizing embryonic culture conditions and thereby increasing the success rates of IVF.

In this context, the objective of this thesis is to understand gene regulatory mechanisms involved in human embryonic development. Considering multiple developmental stages, the central question is how the embryo transitions from one stage to another, and which mechanisms influence these decisions. While various genes are known to be important during certain developmental stages, the regulatory mechanisms between these genes (and potentially unknown ones) remain to be discovered.

This objective presents several challenges. The first challenge is associated to the data. We use single-cell transcriptomic data comprising a vast number of cells and genes to consider (around $1,500$ cells and over $20,000$ genes; see Section 4.2.3.1). This data is noisy and contains a high concentration of zero values, making the analysis complex (see Section 2.2.3). The second challenge concerns the studied system which cannot be perturbed. This complicates the task of modeling and validating the obtained outcomes.

In this thesis, we will focus principally on the trophectoderm (TE) fate maturation. The TE goes through three phases of maturation: early, medium and late (Figure 1.1B). Additionally, the TE, particularly in its late phase, is responsible for the embryo's attachment to the uterus, a crucial role for the future development of the fetus. Therefore, further understanding this maturation and developing computational models of TE will provide invaluable insights for the field.

1.4 Overview of our modeling approach

In this section, we provide the overview of our modeling approach. Consider two developmental stages, for instance medium and late TE, we make the postulate that a medium TE cell can stay in this stage or can differentiate into late TE stage (Figure 1.2A). Medium TE cell can be seen as an initial state, while late TE cell can be seen as a final state. Let us consider this problem as a perturbation problem. We have a perturbation set made on the system and the observations captured at the initial and final states.

Given all possible transitions, the objective is to find paths explaining how the system can go from the perturbations to the observations. State-specific paths will serve to model states. Therefore, both models explain how the system can go from the perturbations to the observations.

Considering single-cell transcriptomic data, each cell had a specific expression for each gene (Figure 1.2B). The idea is to identify similarities and differences, in terms of gene expression, between the two states (or cells). The similarities can be seen as pseudo-perturbations and the differences can be seen as pseudo-observations (Figure 1.2C). Here, we take the example of one cell; however, multiple cells form stages, yielding a set of pseudo-perturbation and pseudo-observation pairs that need to be considered. Afterwards, given all possible gene interactions derived from prior-knowledge, combined to pseudo-perturbations and pseudo-observations extracted for all individual cells at each stage, pseudo-perturbations–pseudo-observations paths are learned, modeling each state (Figure 1.2D).

1.5 Contributions

During this thesis, part of the work I led has been published. Over the past three years, I had the opportunity to be the first author for two publications and contribute as a co-author to a third publication. In addition, a fourth publication, in which I will be the first author, is currently being written for submission to a journal.

The first publication, Bolteau *et al.* [5], presented in Chapter 4, introduces the method we developed aiming to infer Boolean networks from both prior knowledge and single-cell transcriptomic (scRNAseq) data. This work was published in the proceedings of the 19th International Symposium on Bioinformatics Research and Applications (ISBRA 2023). I also presented our work at the conference held in Wrocław, Poland. Our objective in this contribution is to explore human embryonic development, specifically focusing on understanding trophectoderm (TE) maturation. We use scRNAseq data to develop a framework for inferring computational models that distinguish between two developmental stages involved in embryo development. Our method selects pseudo-perturbations from scRNAseq data since actual perturbations are impractical due to ethical and legal constraints. By combining these pseudo-perturbations with prior-regulatory networks, we can infer Boolean networks that accurately align with scRNAseq data for each studied

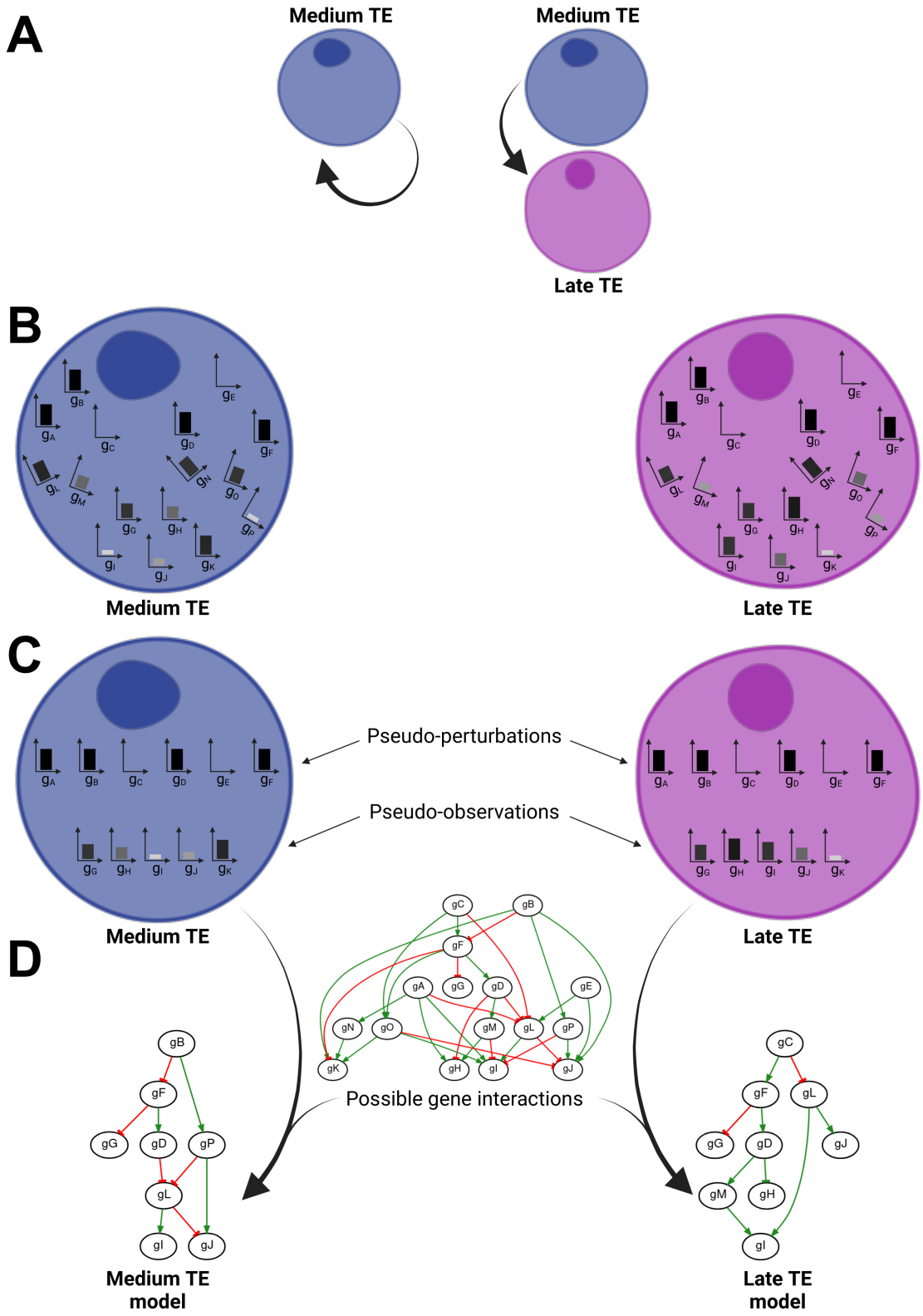


Figure 1.2 – Overview of our modeling approach.

(caption on next page.)

developmental stage. Our publicly available method was tested with several benchmarks, proving the feasibility of our approach. Applied to the real dataset, we infer Boolean network families, corresponding to the medium and late TE developmental stages. Their structures reveal contrasting regulatory pathways, offering valuable biological insights and hypotheses within this domain.

The second publication, Bolteau *et al.* [6], presented in Chapter 4, was published in the Journal of Computational Biology (JCB) in 2024. This contribution provides a deeper exploration of the first paper's findings. Here, we utilize scRNAseq data to model gene regulatory mechanisms involved in two human developmental stages: medium and late trophectoderm (TE). We identify 20 pseudo-perturbations, that are integrated with prior knowledge gene interactions to infer stage-specific Boolean networks (BNs). These BNs delineate distinct regulatory mechanisms, enabling the differentiation between these developmental stages. We show that our program outperforms existing pseudo-perturbation identification tools. Our framework contributes to comprehending human developmental processes and holds potential applicability to diverse developmental stages and other research scenarios.

The third publication, Le Bars *et al.* [7], was published in BMC Bioinformatics journal in 2023. In this publication, where I am a co-author, we introduce MajS, a method aimed at enhancing the modeling of regulatory networks to facilitate their integration with metabolic networks. Given a regulatory network and a discrete partial set of observations as input, MajS tests the consistency between the input data, proposes minimal repairs on the network to establish consistency, and finally computes weighted and signed predictions

(continued from previous page.)
(A) A cell in the medium TE stage can either remain in the medium TE stage or differentiate into the late TE stage.
(B) Each cell exhibits a specific expression pattern for each gene. Some genes are similarly expressed in both cells (e.g., g_A, g_B, g_C, g_D, g_E, g_F), while others show differential expression.
(C) Pseudo-perturbations are identified (g_A, g_B, g_C, g_D, g_E, g_F). Some of other genes form the pseudo-observations (g_G, g_H, g_I, g_J, g_K). Here, we take the example of one cell; however, multiple cells form stages, yielding a set of pseudo-perturbation and pseudo-observation pairs that need to be considered.
(D) Given all possible gene interactions derived from prior-knowledge, combined to pseudo-perturbations and pseudo-observations extracted for all individual cells at each stage, stage-specific models are learned. Activation interactions are represented by normal green arrows ($\longrightarrow$), while inhibition interactions are depicted by red T-arrows ($\dashv$).
Figure created with BioRender.com.

over the network species. We tested MajS by comparing the HIF-1 signaling pathway with two gene-expression datasets. Our results show that MajS can predict 100% of unobserved species. When comparing MajS with two similar (discrete and quantitative) tools, we observed that compared with the discrete tool, MajS proposes a better coverage of the unobserved species, is more sensitive to system perturbations, and proposes predictions closer to real data. Compared to the quantitative tool, MajS provides more refined discrete predictions that agree with the dynamic proposed by the quantitative tool. MajS is a new method to test the consistency between a regulatory network and a dataset that provides computational predictions on unobserved network species. It provides fine-grained discrete predictions by outputting the weight of the predicted sign as a piece of additional information. MajS' output, thanks to its weight, could easily be integrated with metabolic network modeling.

Finally, a fourth publication is currently being written to be submitted to computational biology journal in the near future. This publication will present the ready-to-publish findings of the Chapter 5, showcasing more robust and more relevant Boolean networks modeling medium and late TE stage. This paper will highlight our method, SCIBORG, including enhancements over the method used in the two first published articles.

1.6 Outline of the manuscript

This manuscript is organized around five chapters.

Chapter 2 "State of the art in single-cell transcriptomic data modeling" discusses the current landscape of single-cell data modeling. It includes an overview of single-cell transcriptomic sequencing and an analysis of three modeling tools.

Chapter 3 "Background" presents necessary definitions and provides an overview of the used data. It also reviews the programming paradigm and the various methods used in the implemented study.

Chapter 4 "Contribution 1: Inferring Boolean networks to model human preimplantation development" details the method implemented for modeling developmental stages using Boolean network inference. This chapter highlights the key findings presented in the two published articles Bolteau *et al.* [5] and Bolteau *et al.* [6].

Chapter 5 "Contribution 2: More robust inferred Boolean networks using SCIBORG"

highlights the enhancements we made to the previous method, resulting in more robust outcomes. This second contribution will be the subject of an upcoming article to be submitted to a journal.

Chapter 6 "Discussion and Perspectives" offers an overall discussion of the work presented in this manuscript and outlines future perspectives.

Chapter 2

State of the art in single-cell transcriptomic data modeling

"We create models to have two things that we do not understand, the problem and the model of the problem."

— Unknown author

Summary

This chapter provides a state-of-the-art review of single-cell transcriptomic data modeling in the context of studying human embryonic development. First, we present single-cell transcriptomic sequencing. Next, we discuss two methods commonly used in the field to analyze scRNAseq data. We then introduce three types of methodology of gene regulatory network inference, which are useful for obtaining a global overview of the regulatory mechanisms involved in the studied system. In the final section, we explore methods for modeling biological systems, particularly adapted to the study of cell differentiation. We present three tools and discuss their perspectives on modeling human embryonic development.

2.1 Introduction

In 2002, Hiroaki Kitano stated, "*to understand complex biological systems requires the integration of experimental and computational research – in other words a systems biology approach*" [17]. The use of systems biology has become indispensable for understanding complex biological systems due to the vast amount of data generated by new sequencing technologies, such as single-cell transcriptomic sequencing. The systems biology approach involves collecting systemic data and abstracting it, typically through networks, to represent the studied system. These constructed networks model the system and allow for simulations and predictions.

We can define three principal types of biological networks [18]: *(i)* signaling networks, which comprise interconnected signaling pathways present in cells; *(ii)* gene regulatory networks, which represent the transcriptional interactions of genes within cells; *(iii)* metabolic networks, which encompass biochemical mechanisms involved in cellular functions.

Biological networks can be modeled using various formalisms. Some examples of them are: Bayesian networks [19], Petri nets [20], and Boolean networks [21]. In our project, the large amount of data (scRNAseq data) increases the model complexity. Therefore, our method requires a formalism capable to handle large number of features. We focus on Boolean network formalism due to its strong abstraction, which is easier to manage in modeling. In addition, this formalism has been employed in numerous studies and has proven useful in systems biology modeling [22], reinforcing our choice.

Our thesis falls within the context of model inference aiming to infer, analyze, and understand the regulatory mechanisms involved in a biological system [23]. The modeling is generally based on prior knowledge which can consist on known interactions between genes or sequencing data, *etc.* For our thesis, we focus on gene regulatory networks (GRNs) that incorporate prior knowledge. Diverse methods to reconstruct

GRNs exist, such as: *(i)* inferring GRNs directly from single-cell data, or *(ii)* inferring GRNs from databases containing gene interactions validated by diverse methods, such as experimental or simulated ones. This prior knowledge is refined using observations to build a specific network modeling the system. In our case, we aim to model a cell differentiation phenomenon using single-cell data combined with prior knowledge from databases. The objective of this type of modeling is to understand the gene regulatory mechanisms occurring during cell differentiation.

In this chapter, we first present how single-cell transcriptomic sequencing works. This sequencing method was used to generate the data for our thesis project. In Section 2.3, we present two classical tools used to analyze scRNAseq data, which are utilized in the subsequent methods we present. In Section 2.4, we explore various methods for reconstructing gene regulatory networks from single-cell transcriptomic data. In Section 2.5, we introduce three methods for modeling single-cell data using Boolean networks. Finally, we discuss the modeling methods by outlining their advantages and drawbacks.

2.2 Single-cell transcriptomic sequencing

2.2.1 Revolutionizing cell differentiation studies

For years, bulk RNA sequencing (bulk RNAseq) has been instrumental in understanding biological mechanisms by analyzing pooled cell samples, from specific tissue for instance [24]. After bioinformatics analysis, bulk RNAseq enables the identification of the quantity of genes, i.e., the expression of genes, within a cell population [25]. This technic leads to significant discoveries such as the construction of a map of functional elements in the human genome by the ENCODE consortium [26], or the revealing of transcriptomic signatures in the cortex for Alzheimer disease patient [27]. However, bulk sequencing, while valuable, overlooks the complexities where individual cells impact the global system like in cell differentiation. Thus, another sequencing method is required to better analyze these mechanisms. The introduction of single-cell transcriptomic sequencing (scRNAseq) in 2009 by Tang *et al.* [28] revolutionized cell differentiation studies. By sequencing cells individually, scRNAseq allows for the precise identification of gene expression in each cell and facilitates the discovery of marker genes.

2.2.2 The process of single-cell transcriptomic sequencing

In this section, we describe briefly the process of the scRNAseq starting, in the context of studying human embryonic development, from the embryo to the single-cell gene expression, illustrated in Figure 2.1.

Isolation of cells The first step of scRNAseq is the isolation of cell. In this thesis manuscript, we only focus on embryonic cells sequencing, where cell isolation can differ from others techniques in different domains. Embryos are cultured until reaching the desired developmental stage, followed by laser dissection to separate polar (containing TE, PrE and EPI cells) side and mural side (containing TE cells) [13] (Figure 2.1A and B). Subsequently, cells are dissociated (Figure 2.1C), annotated with essential information like embryo origin or developmental stage, and prepared for sequencing.

Preparation of the library After individual cells are lysed to extract RNA, each RNA molecule is barcoded by attaching short RNA sequences to its ends [29, 30] (Figure 2.1D). These barcodes facilitate the conversion of RNA into complementary DNA (cDNA), necessary for sequencing. cDNAs are then amplified in order to generate a sufficient quantity of molecules for the library preparation and the sequencing [29, 30] (Figure 2.1E). There are several steps in the procedure for preparing cDNA for sequencing, as detailed in Jovic *et al.* [29] and Hwang *et al.* [31].

Sequencing Nowadays, multiple sequencers exist; in this manuscript, we only present the general process of sequencing without presenting the specificity of certain methods. Briefly, the cDNA strand is scanned in order to determine its nucleotide sequence (Figure 2.1F). The resulting sequences, called *reads*, correspond to the transcripts of the cell.

Bioinformatics analysis The reads are aligned to a reference genome in order to identify the corresponding genes, forming a *count matrix* indicating read counts per gene per cell (Figure 2.1G). This (raw) count matrix undergoes various filtering methods to enhance data quality [29, 32, 33]. When integrating datasets from different sequencing runs, normalization ensures comparable gene expression levels. Moreover, this expression can be normalized (for instance, log-normalization) using various tools.

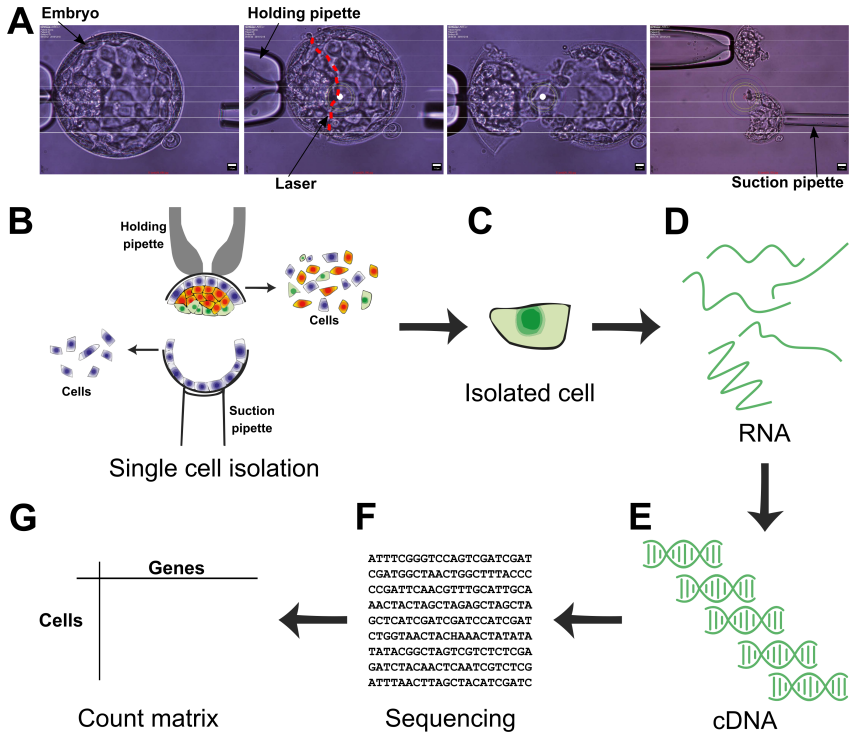


Figure 2.1 – Single-cell RNA sequencing workflow.
(A) Embryo dissection using a laser (red dashed line in second image).
(B) Schematic illustration of the cell dissociation process.
(C) Isolated cells are lysed to extract RNA.
(D) RNA are captured and converted into cDNA.
(E) cDNA are amplified to obtain a sufficient quantity of molecules for sequencing.
(F) After library preparation, cDNA are sequenced, producing multiple reads for each cell.
(G) Bioinformatics analysis process the reads to form a count matrix, comprising the number of reads for each gene in each cell.
Figure adapted from Meistermann et al. *[13].*

2.2.3 Zero-inflation in scRNAseq analysis

An essential consideration in scRNAseq analysis is zero-inflation, wherein a significant proportion of genes exhibit zero expression, a phenomenon more pronounced compared to bulk RNAseq data [34, 35]. Two sources are possible for the zero measurements in scRNAseq data: biological and non-biological [36]. On the first part, biological zero is defined as an absence of a gene's transcripts of messenger RNAs (mRNAs). These zeros arise from two scenarios :

- no transcription, i.e., a non-binding of the RNA polymerase;
- absence of mRNA in the cell due to a faster mRNA degradation.

On the second hand, non-biological zero is due to a loss of information about a truly express gene. Contrary to biological zeros, non-biological zeros are due to a zero-expression measurements of gene with transcripts in a cell. This underscores the need for meticulous attention when analyzing and interpreting scRNAseq data.

2.3 Methods of scRNAseq data analysis

Various methods are employed to analyze single-cell transcriptomic data, aiming to unveil underlying biological mechanisms and formulate hypotheses. These methods use principally statistics or machine learning techniques. Given the vast amount of data produced by scRNAseq, extracting meaningful insights can be challenging. Dimensionality reduction techniques like PCA (principal component analysis) [37] are commonly used to simplify the data. More advanced methods such as t-SNE [38] and UMAP [39] offer complex dimensionality reduction capabilities (see Section 2.3.1).

Additionally, in the context of single-cell analysis, a common biological system studied is cell differentiation. To understand the mechanisms behind these differentiations, one can reconstruct trajectories in order to hierarchize cells and thus form trajectory called *pseudotime* (see Section 2.3.2).

2.3.1 UMAP, a widely used dimension reduction method

UMAP (uniform manifold approximation and projection) [39] is a non-linear technique of dimension reduction, primarly employed for data visualization purposes. It transforms high-dimensional data and produces a low-dimensional graph, facilitating easier visualization and interpretation. The UMAP algorithm comprises two steps (Figure 2.2):

1. **High-dimensional graph construction:** Initially, UMAP constructs a high-dimensional graph that approximates the shape (or topology) of the data, known as the *manifold* (Figure 2.2A). The algorithm constructs a graph where data points are connected to their neighbors based on proximity, with the links weighted to represent connection probability. This process preserves both local and global structure, effectively capturing clusters as high-density regions and separations between clusters as low-density regions.

2. **Low-dimensional projection:** In the second stage, the algorithm projects the high-dimensional graph onto a lower-dimensional space (Figure 2.2B). The algorithm optimizes the positions of data points in the low-dimensional space to retain the local and global structure of the original data. UMAP achieves this by balancing the proximity of nearby points while preserving overall data structure. The aim is to maintain the underlying structure and patterns present in the data, making analysis and interpretation easier.

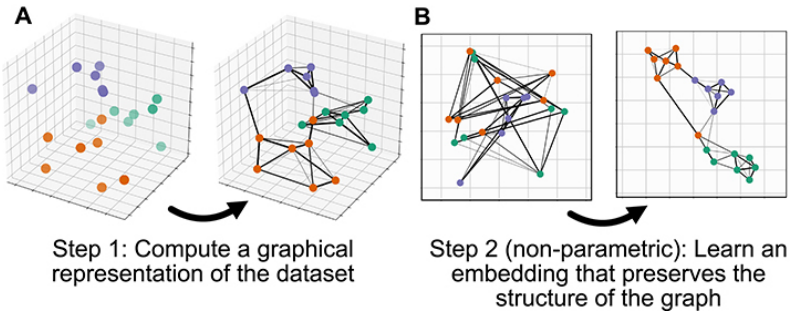


Figure 2.2 – Overview of UMAP.
(A) High-dimensional graph construction.
(B) Low-dimensional projection.
Figure adapted from Sainburg et al. *[40].*

t-SNE (t-distributed stochastic neighbor embedding) [38] is similar to UMAP. Both algorithms build a graph representing data in high-dimensional space and then reconstruct a graph in a lower-dimensional space. In brief, the principal difference between UMAP and t-SNE lies in their approach to reduce dimensions. t-SNE transforms the graph point by point, focusing on preserving the local structure of the data, whereas UMAP considers all points simultaneously by a compressing of the graph, maintaining a balance between local and global structures. Both algorithms are competitive in terms of visualization quality [39, 41]. However, UMAP outperforms t-SNE in terms of runtime performance and its ability to handle larger datasets [41]. UMAP also maintains a balance between local and global structure and produces consistent outputs for the same dataset due to its deterministic nature [41, 42]. In contrast, t-SNE generates different low-dimensional graphs for each run. Additionally, UMAP supports supervised dimension reduction, using the use of categorical label information.

2.3.2 Pseudotime, a cell hierarchy trajectory

To study dynamic process, reconstruct trajectories of cells is an efficient method. The pseudotime is an arbitrary metric measuring the position of a specific cell in the studied dynamic process [43]. By ordering the cells according to this pseudotime one can identify different transition stages in the studied system. Pseudotime can also be conceptualized as the total amount of transcriptomic change from the least "mature" (or root) cell in the cell trajectory [4]. This method offers the possibility to observe the gene expression evolution during a cell transition.

A plethora of method of trajectories inference exists, such as STREAM [44], Slingshot [45] or TSCAN [46]. Most of pseudotime inference methods have similar processes [43, 47], comprising three principal steps: *(i)* a reduction of dimension, *(ii)* a learning of trajectories using tree or graphs, and *(iii)* an assignment of a position on the learned trajectory for each cell.

We propose to explore deeper the methodology of Monocle [48], a software for pseudotime inference used in the paper by Meistermann *et al.* [13], which provides the input results of this thesis (see Section 3.2). Before the reduction of dimension, Monocle allows for the selection of ordering genes, defining the trajectory's progression. These genes can be selected using expert knowledge of the biological data leading to a semi-supervised learning. Otherwise, the software provide tools to select unsupervisedly these genes. Once the ordering genes are selected, Monocle applies the dimension reduction using an algorithm, based on manifold learning, similarly to UMAP (Figure 2.3). Afterwards, Monocle constructs a spanning tree using a set of centroids chosen automatically using a k-means clustering. The positions of cells are then updated based on the current trajectories formed by the tree. Finally, the user selects a "root" meaning the start of pseudotime (value 0). Then, each cell's pseudotime is calculated using a distance along the tree to the root. It is important to note that the cell ordering is "root-dependant" leading to non-unique ordering. For instance, in Meistermann *et al.* [13], which uses Monocle, $5,000$ pseudotime orderings were generated.

2.4 Methods of gene regulatory network inference

In addition to data analysis methods, gene expression in (single-cell) transcriptomic data can be used to infer networks known as *gene regulatory networks* (GRNs). GRNs, mathematically defined as graphs, are generally inferred *de novo* from gene expression

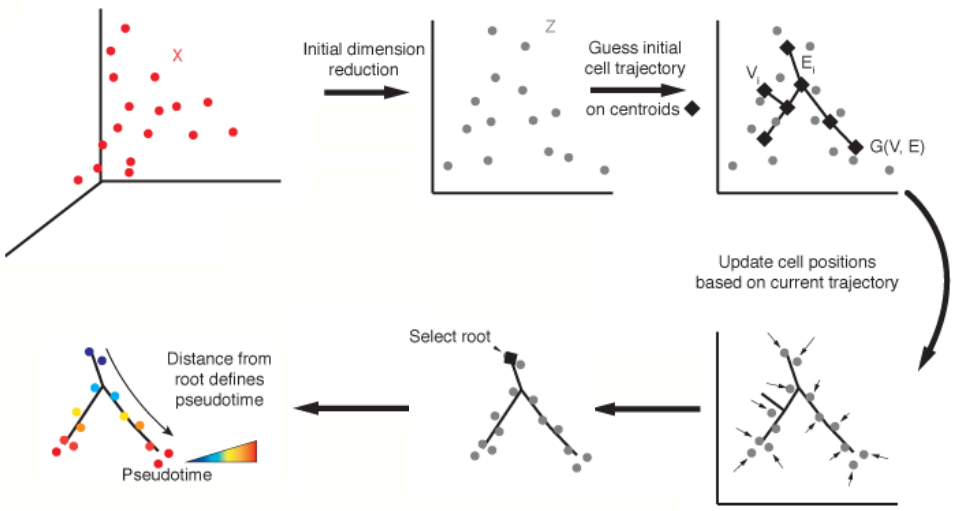


Figure 2.3 – Pseudotime trajectories learning using Monocle.
Each cell is represented as a point in a high-dimensional space where each dimension corresponds to the expression level of an ordering gene. The high-dimensional data (X) are first reduced onto a lower dimensional space (Z). A spanning tree is then constructed selecting automatically a set of centroids. The algorithm moves each cell toward the nearest vertex of the tree. Finally, the user selects a "root" and each cell's pseudotime is calculated based on the learned tree.
Figure adapted from Qiu et al. *[48].*

measurements. A plethora of methods exist for GRN inference from single-cell data, many of which are discussed in reviews by Badia-i-Mompel *et al.* [49] and Nguyen *et al.* [50]. There are three approaches of network construction: *(i)* gene correlation, *(ii)* correlation ensemble over pseudotime, *(iii)* differential equations. All methods follow a similar workflow presented in Figure 2.4. Initially, gene expression data is filtered and then transformed into the necessary structure. Each method employs a specific technique to infer the output network. The resulting GRN can be directed or undirected, and represented as a graph or adjacency matrix, depending on the method. In the following paragraphs, we review these three inference methods, and then discuss them.

2.4.1 Methods review

Gene correlation Gene correlation methods are based on the pair-wise relationship between genes using a correlation metric. First, the methods generate a gene correlation matrix, following a method-specific correlation metric (Figure 2.5i). The generated matrix is often too complex to form a meaningful GRN. Correlation are tested to filter out

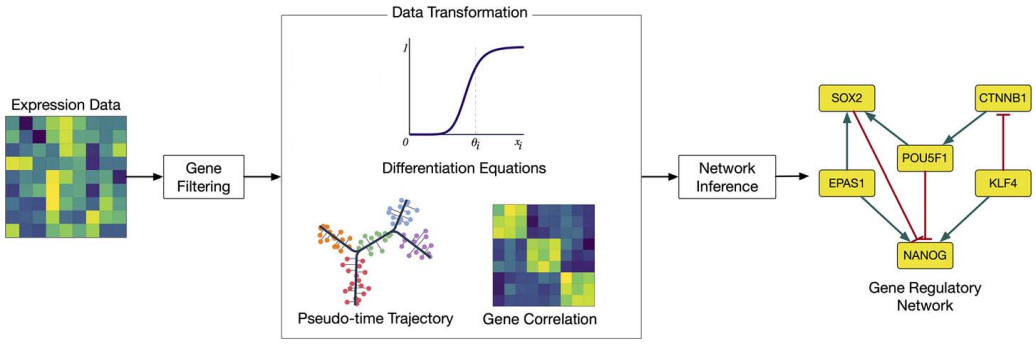


Figure 2.4 – Overall workflow of GRN inference methods.
First, gene expression data is filtered. Data is then transformed, depending on the method, in order to infer a network. This output network can be either co-expression network or directed network.
Figure adapted from Nguyen et al. *[50].*

ones that are not significant (Figure 2.5ii). Finally, kept correlations form the GRN (Figure 2.5iii).

These methods provide a good estimation of the connections between genes but are unable to infer regulations between genes, such as activation of inhibition of one gene on another. Some approaches attempt to overcome this limitation by enriching the network with known regulatory mechanisms from databases. SCENIC [1] [51] examplifies this by combining gene interaction data with transcription factor (TF) binding motif enrichment.

This category of GRN reconstruction is used, for instance, in SINCERA [52], NLNET [53] or Information [54], that are tools to infer GRNs from scRNAseq data using gene correlations.

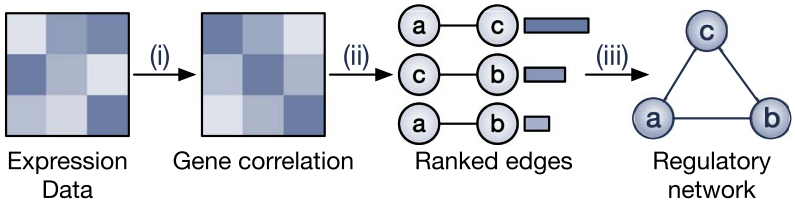


Figure 2.5 – Workflow of GRN inference method based on correlation.
(i) Expression correlations are calculated from for each gene pair.
(ii) Correlations are filtered to keep only significant correlations.
(iii) The GRN is then formed.
Figure from Nguyen et al. *[50].*

1. A technical specification sheet of SCENIC is available in Appendix A.

Correlation ensemble over pseudotime Methods in this category use gene correlation to infer gene relationships. In addition to gene correlation (only) based methods, here, the notion of time is taken into account. These methods assume that the pair-wise relationships may change depending on the developmental stage. Thus, correlations are calculated in specific time windows through a time ordering of cells. The workflow of these methods consist of three main steps. First, the pseudotime trajectories are calculated to order cells (Figure 2.6i). Second, data is divided in small-time windows representing all possible time lags. For each possible time lag, gene correlations are computed, resulting in a series of correlation matrices (Figure 2.6ii). Third, multiple correlations are aggregated into an adjacency matrix indicating the sign of correlation between genes (Figure 2.6iii) This matrix can easily be transformed into a directed and signed GRN. It is important to note that the performance of GRN inference depends on the accuracy of the time ordering, which may be not unique [50].

This category of GRN reconstruction is used, for instance, in methods for network inference such as LEAP [2] [55], SCIMITAR [56], or SINCERITIES [57].

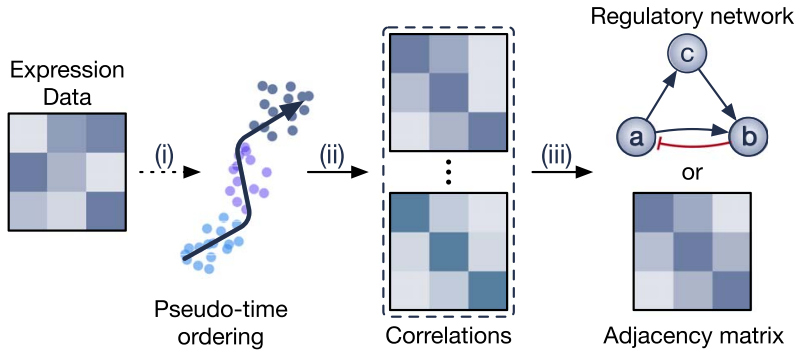


Figure 2.6 – Workflow of GRN inference method based on correlation ensemble over pseudotime.
(i) The pseudotime trajectories are calculated from scRNAseq, providing an ordering of cells.
(ii) Data is divided into small-time windows, and the algorithm calculates the gene correlation for each time window.
(iii) The method merges multiple correlation matrices into one adjacency matrix, which can be then transformed into a GRN.
Figure from Nguyen et al. *[50].*

2. A technical specification sheet of LEAP is available in Appendix B.

Differential equations Differential equations methods describe gene expression dynamics over time using differential equations. They require time ordering of cells from gene expression data (Figure 2.7i). The methods define differential equations that describe the relationships between genes with the respect to the inferred time (Figure 2.7ii). Parameters used in these equations are estimated to finally yield an adjacency matrix of the gene correlations (Figure 2.7iii and iv). Given the adjacency matrix and temporal information, the methods can infer directed and signed GRN. The parameter calculation through the differential equation is dependent to the number of genes and cells The computation complexity and the number of parameters will exponentially increase with large datasets. Therefore, this method category is more suitable for small-size networks [50].

This category of GRN reconstruction is used, for instance, in SCODE [58], SCOUP [59], or Inference Snapshot [60].

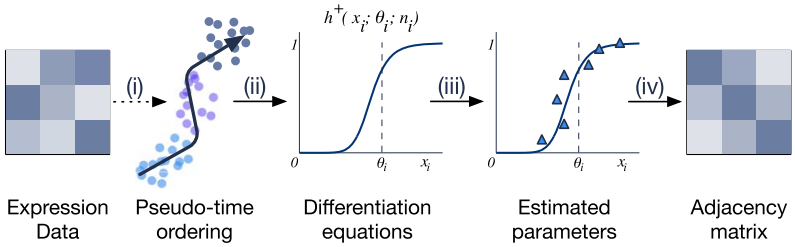


Figure 2.7 – Workflow of GRN inference method based on differential equation.
(i) From pseudotime trajectories inferred from scRNAseq data, differential equations are calculated to describe the relationships between genes.
(ii) The parameters used in the equations are estimated using different optimization techniques.
(iii) The relationships between genes are inferred using the optimized parameters, leading to an adjacency matrix of GRN.
Figure from Nguyen et al. *[50].*

2.4.2 Discussion

We present three families of methods using different frameworks for constructing GRNs. These categories, compared in Table 2.1, all use scRNAseq data as input. The first category, enables the reconstruction of GRNs using only gene correlation without direction or sign on the edges. This issue can be addressed with enrichment data, as demonstrated by the SCENIC tool. Enrichment data refers to external sources of information that are combined with internal data, in our case scRNAseq data, to improve its precision

Table 2.1 – Comparison of gene regulatory network inference methods.

Method category	Inputs	Directed GRNs	Signed GRNs
Gene correlation	• scRNAseq data • (Binding site*)	✗*	✗*
Correlation ensemble over pseudotime	• scRNAseq data • Pseudotime trajectories	✓	✓
Differential equation	• scRNAseq data • Pseudotime trajectories	✓	✓

* By incorporating enrichment data using certain tools, it is possible to infer directed and signed GRNs.

and accuracy. Enrichment allows the inference of directed and signed networks; however, the available information is limited. The two others categories, correlation ensemble over pseudotime and differential equation methods, allow the inference of directed and signed GRNs. In addition, we note that the differential equation method is limited by the dataset it can handle. The complexity of the method is correlated with the number of parameters, which imposes a constraint on its applicability to larger datasets.

2.5 Methods of modeling

In the previous sections, we presented various methods for analyzing single-cell transcriptomic data and inferring GRNs. The observations derived from these methods provide a detailed picture, helping us to understand the systemic view of the studied system based on the data. However, these methods have limitations when it comes to generating predictions in cases of system modifications, such as genetic perturbations or environmental changes. Therefore, it is crucial to delve deeper into the analysis of scRNAseq data to overcome these limitations. In this section, we introduce three methods used to model cellular differentiation processes. Following this, we discuss the applicability and implications of these methods for modeling human embryonic development.

2.5.1 Methods Review

BoNesis, a synthesis tool of ensembles of Boolean networks The first method we discuss is BoNesis [61, 62], which automatically synthesizes dynamic models represented by ensembles of Boolean networks (BNs). This approach frames modeling as a Boolean

satisfiability problem encoded in ASP, allowing the inference of BNs from the dynamical properties of biological processes. BoNesis combines experimental data with prior knowledge, treated as constraints, to generate ensembles of BNs compatible with both. By integrating dynamic processes, this tool enables researchers to model phenomena such as cell differentiation. The prior knowledge and biological observations are incorporated into a logic program. Given defined constraints and rules, this logic program is then solved yielding solutions composed of BNs that include: *(i)* gene interactions from all possible ones given the knowledge, and (ii) dynamical properties compatible with observation behaviors.

BoNesis requires three principal inputs: the prior knowledge network (PKN), biological observations and dynamical properties. The PKN is an interaction graph comprising components, particularly genes, and interactions characterized by activation and inhibition between components (see Section 3.1.1 for interaction graph definition). The PKN delimits the interaction that can be used by the synthesized BNs. Biological observations provide information on the evolution of components in the studied process. They can be gene expression measurements from transcriptomic sequencing, for instance, associated with different time points (of an experiment or a cell). BoNesis works only with binarized observations, necessitating a binarization step between the data collection and BoNesis usage. The last input, dynamical properties, are discrete-time observations of the system representing the behavior to be reproduced.

In addition to BN synthesis, BoNesis offers two complementary functionalities. The first is the diversity in enumeration of the models. It is common to have numerous compatible solutions for a given problem. Thus, BoNesis identifies a sub-ensemble of possible models. This sub-ensemble should be the more relevant possible regarding the diversity of BNs. While ASP solvers can enumerate compatible solutions by default, they often explore solutions with minimal variations, leading to strongly similar solutions. To address this, BoNesis employs heuristics of the solver to explore distant solutions. The second functionality is the component selection. Typically, a large PKN includes many components, but only a few are involved in the biological process. BoNesis defines an optimization criterion to maximize the number of specific components, known as "strong constants", within the compatible BN. A strong constant is a component with a constant value necessary for reproducing observations within the BN dynamics. By prioritizing these strong constants, BoNesis ensures compatibility with biological data while simplifying the interaction domain to focus on key regulating interactions of the

observed process.

In Chevalier [63], the author uses BoNesis to model the regulation in hematopoiesis, the differentiation process of blood cells. scRNAseq data from Nestorowa *et al.* [64] were used to model this differentiation phenomenon. First, single-cell data were preprocessed in order to extract observations and determine dynamical properties. Pseudotime trajectories were reconstructed, revealing two bifurcations leading to three cell fates (Figure 2.8). Observations were defined from trajectories by assembling neighboring cells at key steps such as root, bifurcations or leaves (Figure 2.8, black circles). In total, six steps were identified ($S0$ to $S5$), each comprising tens to a hundred of cells. Then, gene expression were binarized using PROFILE method [8], which determine thresholds for each gene based on its value distribution among cells, resulting in a presence of the gene (1), absence (0), or non-significative expression (NA). All cells of the dataset were binarized, producing a matrix composed of 0, 1 or NA. The expression value for the six observations was then determined by setting the majority value of the constituent cells. Second, dynamical properties were defined, such as positive reachabilities (e.g., allowed paths from $S1$ to $S5$ through $S0$ and $S3$ for instance), negative reachabilities (e.g., impossibility to go from $S2$ to $S3$), or fix points (e.g., leaves like $S4$). Third, the PKN was constructed from the DoRothEA database [65]. BoNesis was used to select relevant components from the reconstructed PKN, thereby reducing its size. Finally, BoNesis synthesized $1,000$ BNs, all compatible with the PKN and the observations.

Figure 2.8 – Pseudotime trajectories of blood cells.
Pseudotime trajectories inferred from the STREAM tool [44]. Identified observations, represented by circles, comprise a set of neighboring cells at key steps of the trajectories.
Figure from Chevalier [63].

RE:IN, a method to synthesize logical models RE:IN (Reasoning Engine for Interaction Networks) [66, 67] is a tool designed for the synthesis and analysis of logical models to understand biological processes, such as cellular decision-making. Given a set of experimental data and potential interactions, RE:IN constructs Boolean network models, using an automated reasoning, via SMT (Satisfiability Modulo Theories) encoding. Rather than generating a single network model, RE:IN produces a set of models, each consistent with the experimental data and possible behaviors, enabling researchers to make predictions and explore various biological questions.

RE:IN requires two inputs: an abstract Boolean network (ABN), and experimental constraints. First, the ABN defines potential interactions between genes in the studied system (Figure 2.9). In addition to the genes, inputs conditions are considered, enabling the perturbations on the system. These possible interactions are inferred from gene expression correlation extracted from biological experiments. Second, the experimental constraints defined the setup of diverse leaded experiments (Figure 2.9). Given a perturbation on the inputs and network components (such as an activation, an inhibition or a knockout) the behaviors of the genes is captured. Thus, for each constraint, there is an initial observation and a final observation. Combined the experimental constraints and the ABN, the algorithm computes constrained ABNs (cABNs) that are ABNs consistent with experimental constraints, encompassing all possible mechanisms that match the observed system behavior (Figure 2.9). An asynchronous update strategy is employed to identify the sequence of activation of each component. Thus, computed cABNs formulate the required interactions to model the studied system. The set of consistent cABNs can be enumerated, enabling the identification of diverse mechanisms. Additionally, the minimal cABN can be highlighted to show interaction critical for the network. cABNs can also be used to formulate predictions, allowing researchers to determine if new hypotheses are supported by the cABNs. Once these predictions are tested experimentally, they can be incorporated into the set of experimental constraints and potentially refining the cABNs.

In Dunn *et al.* [68], the authors applied RE:IN to derive Boolean networks models of the naïve pluripotency maintenance and induction. The aim was to study mouse post-implantation epiblast stem cells (EpiSCs) and their resetting into a pluripotent state, which gives EpiSCs the capacity to differentiate into other cell types. The inferred Boolean networks captured behaviors of transcription factors and gene activation trajectories during this process. They generated $10,000$ models and identified the "minimal model" which is the cABN having the fewest interactions and constrained with experimental

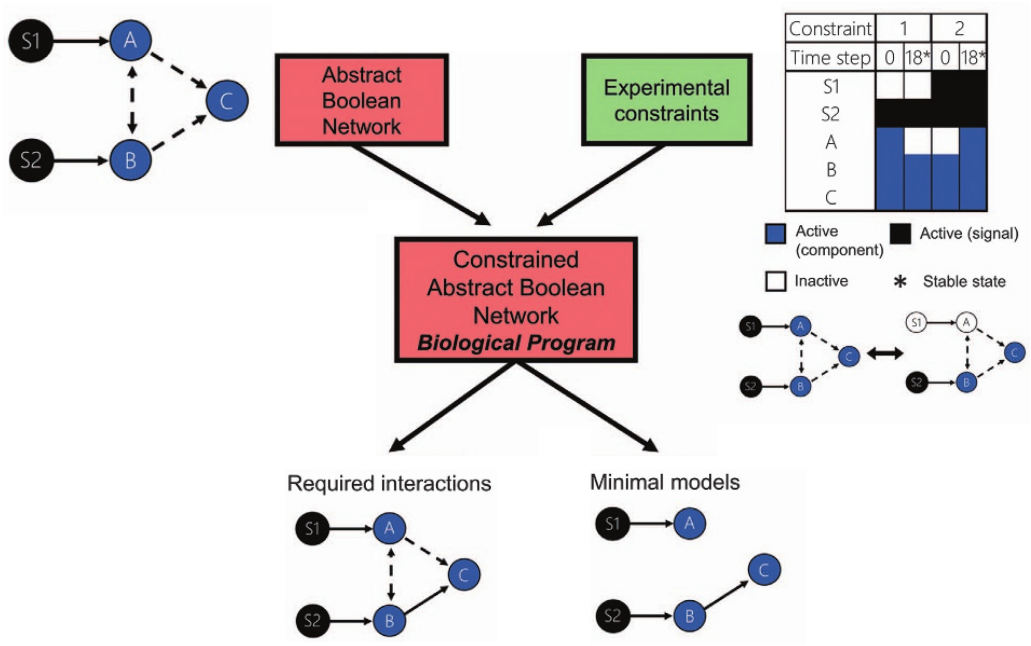


Figure 2.9 – RE:IN methodology.
Combining an abstract Boolean network (ABN) and experimental constraints, constrained ABN (cABN) are inferred. These cABNs formulate the required interactions and the minimal cABN can be generated. *Figure adapted from Yordanov* et al. *[67].*

gene perturbation data, to model the dynamic behavior of cell differentiation. Moreover, the authors used cABNs to formulate predictions on the constraints needed to reset EpiSCs into a pluripotent cell state. They identify specific constraints, such as activation, inhibition or knockdown of transcription factors, to enable EpiSCs to reset.

SCNS, a toolkit for synthesizing single-cell networks SCNS (Single-Cell Network Synthesis) [69] toolkit generates computational regulatory network models. Given single-cell binarized expression states, it learns Boolean networks (BNs) exhibiting the on/off patterns of transcription factors (TFs) [69]. The model learning process is framed as a Boolean satisfiability problem encoded in SMT (Satisfiability Modulo Theories). Notably, the network construction solely relies on single-gene transition deduced from the single-cell data.

SCNS begins with dimension reduction based on the concept of diffusion distance, constructing a diffusion map (Figure 2.10A). Briefly, this algorithm calculates the similarities between cells based on their gene expression patterns. The resulting low-dimensional map reflects cell similarities and represents the developmental trajectories

over time. Similarly to pseudotime, the diffusion map provides a cell ordering in developmental time, enabling identification of potential bifurcations. Afterwards, gene expression is binarized, forming binary states. A state graph is built from binarized single-cell data (Figure 2.10B). In this graph, states are connected when the expression of one gene differs between two states, resulting in a connected state transition graph comprising single-gene transitions. This graph represents the possible developmental expression state changes and serves as the foundation for BNs learning. Since the direction between two states in the graph is not predefined, the algorithm determines the orientation of edges within the transition graph by using the order of cells derived by the diffusion map analysis. For that, users specify the initial and final states, which correspond to the "root" stage and the "leaves" of the differentiation process, respectively. Thus, for each gene, Boolean update functions consistent with all state transitions are determined. Finally, the resulting networks form the BNs that model the studied system (Figure 2.10C).

In Moignard *et al.* [69], the authors applied SCNS to analyze the early blood cellular differentiation with the goal of better understanding the mechanisms involved in cell progression. They focus on approximately 40 transcription factors involved in the blood cell differentiation and consider the expression of these genes in 3,934 cells. A diffusion map was computed, enabling the identification of developmental trajectories and the ordering of cells. Subsequently, they compute the transition graph comprising around 1,500 expressions states, which was used to infer Boolean networks. From the various combinations of rules, more than 46,000 Boolean models were generated. The study not only identified previously known transcription factor interactions but also discovered the roles of two transcription factors through predictions, which were then confirmed experimentally.

2.5.2 Discussion

We previously presented three methods for modeling scRNAseq data. The first method, BoNesis, synthesizes ensembles of Boolean networks. Inferred BNs dynamically model a biological system and are particularly suited for studying cellular differentiation. Using these logical models, researchers can gain a global overview of the studied system. In addition, BoNesis handles observations by selecting neighboring cells at key events in pseudotime trajectories, with the average sign in this cell group determining the observation sign.

The second method, RE:IN, also uses BNs to model biological systems, such as cellular

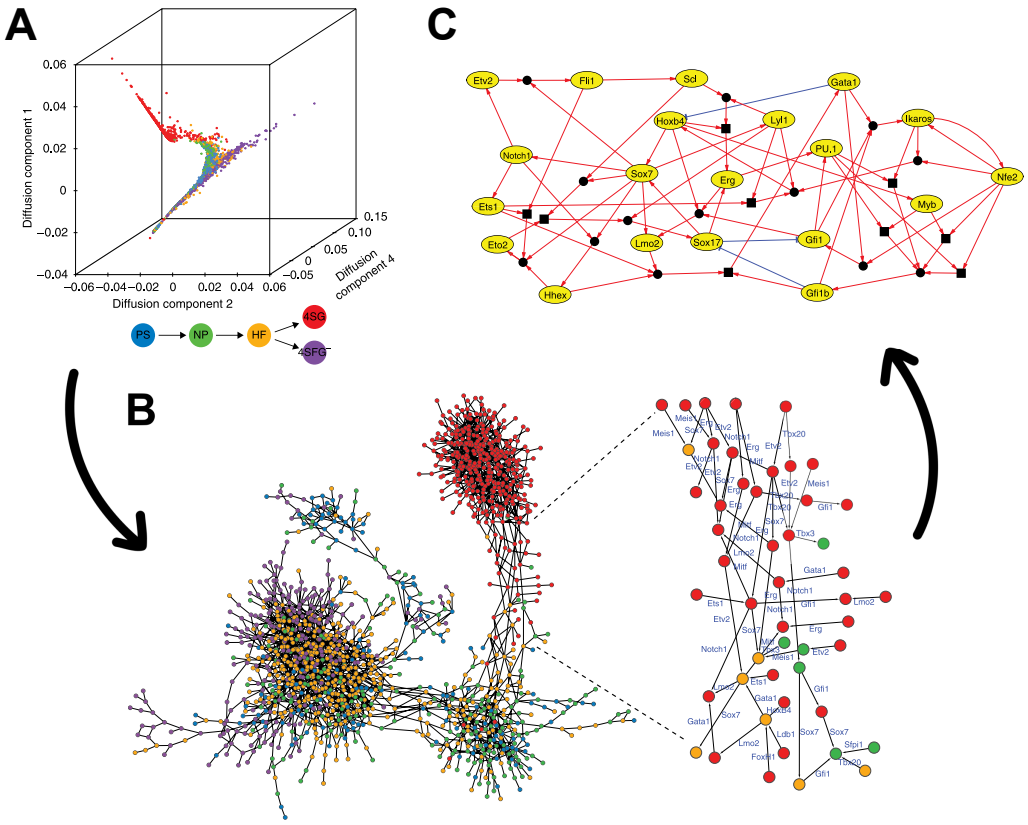


Figure 2.10 – SCNS methodology.
(A) Diffusion map identifying developmental trajectories.
(B) State graph of 1,448 expression states.
(C) Boolean network model. Red edges represent activation, while blue edges represent inhibition. Dark squares represent "AND" gates, while circle indicate "OR" gates.
Figure adapted from Moignard et al. *[69].*

decision-making. The method requires a prior knowledge and experimental constraints, which are biological observations from experiments, such as activation of a component. These inputs enable the synthesis of set of logical models, which can make predictions and suggest new experiments to refine the experimental constraints.

The third method, SCNS, models cell differentiation by using diffusion maps to order cells in time and learn BNs. The method generates state transition graphs enabling the identification of precise transcriptional regulatory mechanisms involved in the studied system.

We compare these methods in Table 2.2. All three methods require binarized gene expression values, which might lead to a strong abstraction, potentially missing interesting

Table 2.2 – Comparison of modeling methods.

Method	Input content	Input size	Prior knowledge	Cell heterogeneity	Cellular dynamic evolution	Exhaustive enumeration	Validation
BoNesis [61]	• scRNAseq • PKN • Dynamical properties	• ≈ 1,000 genes • ≈ 600 cells	• Potential gene interactions • Dynamical constraints	✗	✓	✗	✗
RE:IN [66]	• ABN • Perturbation observations	• ≈ 20 genes • ≈ 30 perturbations	• Potential gene interactions • TFs implicated in the system	✗	✗	✗	✓
SCNS [69]	• Single-cell transcriptomics	• ≈ 40 genes • ≈ 4,000 cells	• TFs implicated in the system	✓	✓	✓	✓*

* Thanks to experimental perturbations.

information. However, this data processing is widely used in systems biology and has proven useful [22]. BoNesis and SCNS require single-cell transcriptomic data, whereas RE:IN uses perturbation observations as input. Additionally, BoNesis and SCNS need cell ordering derived from pseudotime trajectories. All the methods necessitate prior knowledge. For BoNesis and RE:IN, this includes a set of potential gene interactions, which define the interactions used to infer BNs and take the form of a PKN and an ABN, respectively. BoNesis also requires dynamical constraints based on the knowledge of the studied system. Moreover, SCNS and RE:IN need a set of transcription factors (TFs) implicated in the studied system. Regarding input size, RE:IN handles small datasets (approximately 20 genes and 30 experimental perturbations). SCNS can process a similar number of genes but manages a much larger number of cells (around 4,000). BoNesis can accommodate a larger number of genes, around 1,000. Furthermore, BoNesis employs ASP, while the two other methods use SMT.

All three methods aim to learn optimal models that satisfy the given constraints. SCNS focuses on a small evolutionary window of the studied system, enabling it to manage cell heterogeneity, through the construction of a stage graph. We define cell heterogeneity as the consideration of diverse gene expression states across all cells. RE:IN cannot handle cellular dynamic evolution, while the two others methods can model the cell differentiation dynamics of the system. In addition, all the three methods infer families of Boolean network models. These model enumerations are exhaustive in theory, but depending on the total search space and the constraints imposed, it is not always feasible. This is the case of BoNesis and RE:IN, where only a subset of BNs was used (1,000 for BoNesis [63] and 10,000 for RE:IN [68]). SCNS provided a complete enumeration, sometimes resulting in very large sets (46,646 BNs [69]). When no experimental validation data can be obtained,

it is difficult to prioritize which models to favor. Even when punctual (non-exhaustive) experimental data is available, it is important to develop metrics that identify robust BNs compared to other possible BNs that satisfy the constraints. This robustness study was presented in our work, where millions of BNs were possible to be enumerated.

Let us put the three methods into perspective with the thesis, which aims to understand and model mechanisms involved in the human preimplantation embryo development. Our goal is to describe underlying gene regulatory mechanisms implicated in the developmental stages. In addition, within the context of the research project, the global objective is to define the dynamic processes involved in the cellular differentiation, leading to the three cell fates: epiblast, primitive endoderm and trophectoderm.

Given this context, the BoNesis method's approach of analyzing subsets of cells might exclude some cells of the same type, potentially omitting valuable gene expression data. Additionally, calculating the observation sign from an average of cells might dilute unique gene expression patterns, potentially missing interesting behaviors.

The RE:IN method's input requirements are restrictive, relying on constraints from biological experiments involving activation or inhibition. In the context of human embryos, these system perturbations are infeasible due to various factors, including legal ones.

The inference of BNs from the transition graph used by SCNS is highly complex due to combinatorial possibilities involved. For small datasets, such as the one containing around 40 genes in Moignard *et al.* [69], computation remains feasible via an SMT solver. However, SCNS cannot handle larger datasets, making it impractical for our project due to the abundance of data we have (around $1,700$ cells and $20,000$ genes; see Chapter 3 for more details).

Ultimately, these three tools, though very interesting to model embryonic development data, have some characteristics we aimed to approach differently in this work. BoNesis remains interesting for addressing the second objective of the research project, which aims to understand the dynamic processes of the system. However, in this thesis, we focus on the modeling of gene regulatory mechanisms involved in the human embryonic development. To achieve this, we implement a method that provide models of regulatory mechanisms in developmental stages, allowing us to better understand how cells transition from one stage to another and commit to specific cell fates. Additionally, our method offers an exhaustive enumeration and optimality of models while accounting for cell heterogeneity and redundancy. The inferred models are evaluated using a robustness metric, increasing

our confidence in the obtained results.

Chapter 3

Background

"Logic will get you from A to B. Imagination will take you everywhere."
— Albert Einstein (1879-1955)

Summary

This chapter offers various definitions essential for understanding the manuscript. It provides an overview of the data and introduces answer set programming (ASP) through an explanation of the paradigm and a toy example. Additionally, it presents some studies and tools we use in our modeling method.

3.1 Definitions

3.1.1 Prior Knowledge Network

Definition 3.1 Interaction Graph (IG). *An interaction graph $G = (V, E, \sigma)$ is a signed and oriented graph, where $V = \{v_1, \ldots, v_n\}$ is the set of nodes, $E \subseteq V \times V$ is the set of directed edges, and $\sigma \subseteq E \times \{+1, -1\}$ is the signs of the edges.*

Figure 3.1 presents an example of an IG, composed of 6 nodes and 8 edges. In the context of gene regulation, a node represents a gene (or biological entity), and an edge denoted by $j \to i$ indicates that the change in activity level of the gene j influences the activity level of gene i. Edges $j \to i$ are labeled with a sign, where $+1$ (resp. -1) indicates that j tends to increase (resp. decrease) the level of i. We represent an increasing or activation as a classical arrow in green ($\longrightarrow$, Figure 3.1) whereas decreasing or inhibition is represented by a red "T-arrow" ($\longrightarrow\!\!\!\!\dashv$, Figure 3.1).

Definition 3.2 Prior Knowledge Network (PKN). *A Prior Knowledge Network is an IG derived from prior regulatory knowledge, where nodes correspond to biological entities (in our case, genes), and edges represent causal or functional relationships between these entities.*

Within a PKN, we define three types of genes:

- the *input gene*, which is a gene without any predecessor;
- the *intermediate gene*, with predecessor(s) and successor(s);
- the *readout gene*, without any successor.

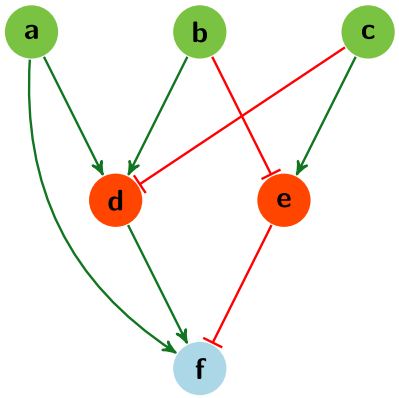


Figure 3.1 – Example of an interaction graph.
This interaction graph comprises 6 nodes and 8 edges. We denote 3 different nodes: inputs in green, intermediates in orange and readouts in blue. Two types of interactions between nodes are present: positive interactions or activations represented by a green arrow, and negative interactions or inhibitions represented by a red "T-arrow".

Input and intermediate genes correspond to the part of the PKN that can be stimulated (externally or internally), they can also be referred to as system *entries*. While readouts are the part of the system that can be observed, they can be referred to as the system *output*. Referring to Figure 3.1, we observe 4 inputs in green, 2 intermediates in orange and 1 readout in blue, along with 3 activations and 3 inhibitions.

3.1.2 Boolean Network

Definition 3.3 Boolean Network (BN). *A Boolean network B, of dimension n is defined as $B = (N, F)$ where:*

- $N = \{v_1, \ldots, v_n\}$ *is a finite set of nodes (variables or genes);*
- $F = \{f_1, \ldots, f_n\}$ *is a set of Boolean functions* $f_i : \mathbb{B}^n \to \mathbb{B}$, *with* $\mathbb{B} = \{0, 1\}$, *describing the evolution of variable* v_i.

Each Boolean function f_i defines the state of a variable v_i based on the states of other variables. Boolean expressions are used to describe the values of these Boolean functions. These expressions comprise logical components, including "AND" ($\wedge$), "OR" ($\vee$), and "NOT" ($\neg$), along with binary Boolean variables that can take values of 0 or 1. They can be represented in either conjunctive normal form (CNF) or disjunctive normal form (DNF).

A CNF is a conjunction of clauses, where each clause is a disjunction of Boolean variables or their negation. Similarly, a DNF is a disjunction of clauses, where each clause is a conjunction of Boolean variables or their negation. Notably, any CNF can be expressed as a DNF, and vice versa. For instance, a CNF $\mathcal{C}$, where $\mathcal{C} = (\neg x_a \vee x_c) \wedge (x_b \vee x_c)$, such that $(\neg x_a \vee x_c)$ and $(\neg x_a \vee x_c)$ are disjunctions, can be written as a DNF $\mathcal{D} = (\neg x_a \wedge x_b) \vee x_c$.

In this manuscript, DNFs are used. Additionally, Boolean functions can be graphically represented using a directed hypergraph, where nodes depict Boolean variables and directed hyperedges describe logical interactions.

In summary, a Boolean network uses Boolean functions to model the interactions and state transitions of variables or genes, with each function defined by a Boolean expression composed of logical operators and binary variables.

Figure 3.2A presents a BN B, of dimension 4, meaning that it is composed of 4 Boolean functions associated to a network's component. Here, we define x, the vector comprising all the studied components: $x = \{x_a, x_b, x_c, x_d\}$.

For instance, the function associated to the first component a is determined by its own activation. Thus, $f_a(x) = 1$ if and only if, in input, the component a is set to 1. Biologically, this local function can be seen as an auto-activation of the component a. Furthermore, the second component, b, is associated with component a, particularly by its negation. Unlike $f_a(x)$, $f_b(x) = 1$ if and only if, in input, the component a is set to 0.

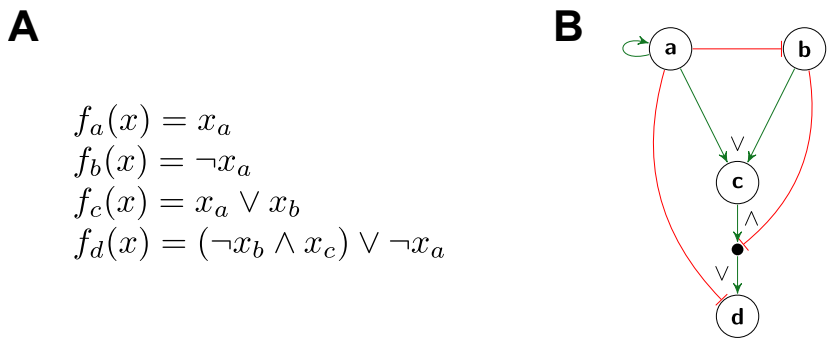


Figure 3.2 – Example of a Boolean network and its hypergraph representation.

(A) Example of Boolean network B of dimension 4.
(B) Directed hypergraph representation of the BN B. Positive edges are represented in classical green arrows ($\longrightarrow$), while negative edges are represented in red "T-arrows" ($\dashv$). Logical interactions are represented with a $\vee$ for a "OR" logic gate, whereas "AND" one are represented with a dark node, denoted with a $\wedge$.

In Figure 3.2B, we depict the directed hypergraph of the BN B shown in Figure 3.2A. The four nodes in the hypergraph represent Boolean variables, each capable of taking values of either 0 or 1. Directed hyperedges, denoting interactions, can be either positive, indicated by an arrow ($\longrightarrow$), or negative, represented by a "T-arrow" ($\longrightarrow\!\!\!\!\!\!\dashv$). Dark nodes ($\bullet$) denote a $\wedge$ ("AND") interactions or gates, while the absence of dark node represents a $\vee$ ("OR") interaction.

For the remainder of the manuscript, BNs will be represented using hypergraphes. Furthermore, the logical symbols ($\wedge$ and $\vee$) will be omitted to streamline the visualization. If there is no dark node between interactions (signifying an "AND"), an "OR" interaction will be assumed.

In this first section, we defined the essential concepts required to understand our implemented method. We will now present the single-cell data used in our method.

3.2 Single-cell transcriptomic data from human embryos

3.2.1 Composition of the data

Throughout this thesis, we exploit scRNAseq data obtained from stage-matched human embryos, leveraging the dataset initially compiled by Petropoulos *et al.* [70] and subsequently refined in Meistermann *et al.* [13]. This dataset encompasses the expression profiles of $34,054$ genes across $1,496$ cells derived from 88 stage-matched human embryos. Notably, scRNAseq data often exhibits zero-inflation (see Section 2.2.3), a phenomenon we encounter in our dataset, where 63% of the values are zeros.

3.2.2 Data analysis

The study Meistermann *et al.* [13] delineates cell fate decisions during the early human embryo development. Three principal analyses were conducted to unravel cell fate transcriptional signatures and the hierarchy of events occurring during the development.

UMAP clusters WGCNA [1] (weighted gene co-expression network analysis) [71] was employed to identify gene expression signatures associated with distinct developmental stages and lineages. This method identifies module eigengenes representing a linear combination of gene subset activation or repression. These modules are then used to produce a precise map of the transcriptome identity, via dimensionality reduction using the UMAP (uniform manifold approximation and projection) approach [39]. Cells are grouped (unsupervisedly) into 8 major clusters indicative of a lineage or a stage (Figure 3.3A). Other clusters are qualified as "undefined" or "intermediate".

Pseudotime Single-cell trajectories were reconstructed using Monocle 2 software [48]. Cells are ordered along a pseudotime, placing them along a trajectory corresponding to a biological process such as cell differentiation by leveraging the asynchronous progression of these processes in individual cells. This method enables the identification of cell fate decisions and their regulated genes using machine learning techniques. Figure 3.3B depicts the inferred pseudotime and the cell hierarchy, facilitating the understanding of the temporal evolution of cells during the human preimplantation embryonic development. This analysis reveals two specifications leading to three cell fates: epiblast (EPI), primitive endoderm (PrE) and trophectoderm (TE).

Lineage signature By combining gene module expression with each cell cluster obtained with the UMAP, 8 modules of genes can be identified (Figure 3.3C). The heatmap reveals module-specific behaviors, providing insights into the potential roles of genes within each module. Some modules are related to specific lineages, such as the GATA4 module for PrE, the GATA2 for (early, medium and late) TE and the NR2F2 module for late TE. Conversely, some modules reflect overall change in embryo development, such as DUXA module, which is associated with zygotic genome activation (i.e., the initiation of gene expression after fertilization). Moreover, authors identify a list of 438 transcription factors (TFs) implicated in these 8 modules, where some of them are indicated on the right of the heatmap of the Figure 3.3C. The complete list of TFs can be retrieved in Appendix D.

1. A technical specification sheet of WGCNA is available in Appendix C.

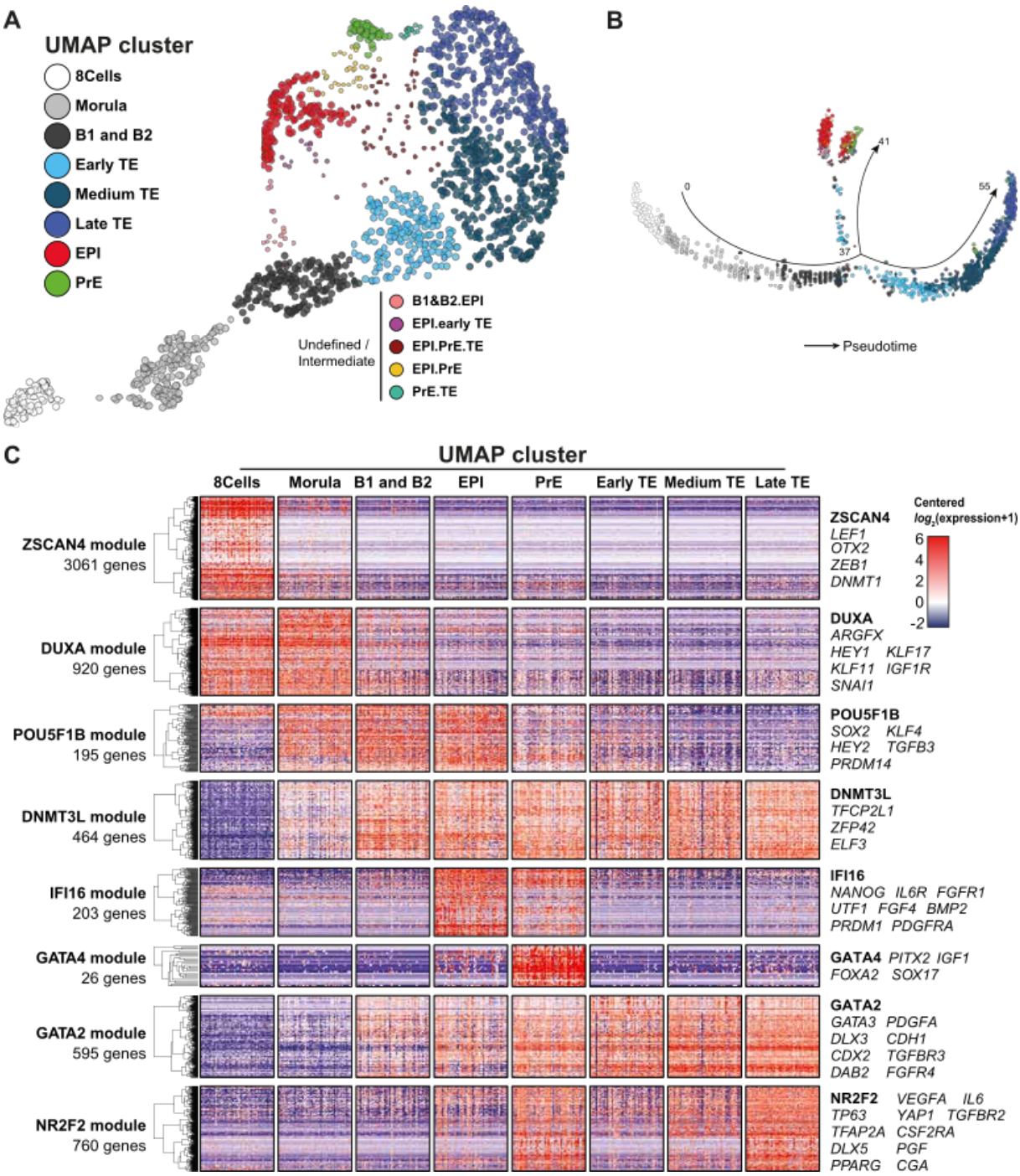


Figure 3.3 – Single-cell transcriptomic data representation.

(A) Dimension reduction (UMAP) of WGCNA module eigengenes, colored by density-based clusters. Small dots indicate a minor cluster composed by the intermediate transcriptome. Medium TE and late TE were segregated by additional k-means clustering based on NR2F2 and GATA2 module eigengenes (see Meistermann *et al.* [13] for more details).

(B) Projection of density-based clusters from (A) on the pseudotime.

(continued on next page.)

3.3 Answer set programming

3.3.1 Presentation of the paradigm

Logic programming (LP) is a programming paradigm based on formal logic [72]. LP defines applications using a set of facts, logic rules and constraints. LP allows programmers to write programs and solve them [73]. Two interpretations of a logic program can be done: *(i)* the declarative interpretation, which refers to what is being computed and *(ii)* the procedural interpretation which explains how the computation takes place [73]. Answer set programming (ASP) [74] is a logic programming paradigm adapted to combinatorial search problems, primarily NP-hard search problems [75]. ASP is based on the stable model semantics of LP. The aim is to use programs computing and searching stable models, respecting the rules and constraints given by the program [76].

ASP can be applied to several areas of science. For example, Nogueira *et al.* [77] developed a system able to solving some planning and diagnostic tasks related to the operation of Space Shuttle. In music composition, an automated system implemented in ASP can compose rhythmic, melodic and harmonic music, serves as a computer-aided composition tool and diagnoses errors in human compositions [78]. ASP is also heavily used in biology and systems biology. Brooks *et al.* [79] have inferred and reconstructed a phylogeny for a set of taxa using ASP. ASP is particularly used in systems biology, for instance, for Genome-Scale Metabolic Network (GSMN) reconstructions [80], or metabolic network completion [81]. Other studies use ASP to predict weighted unobserved nodes in a regulatory network [7].

3.3.2 Basic ASP syntax

Term A term is the smallest block in ASP. It can be an integer, a constant, a string or a variable. Constants and variables are distinguished by their initial letters, with constants starting with lowercase letters, e.g., `penguin` and variables starting with uppercase letters,

(continued from previous page.)
(C) Expression heatmap of all genes related to WGCNA modules. Each row represents a WGCNA module, and each column is a set of 50 cells drawn from a UMAP cluster (A). The height of each row represents the number of genes in the corresponding WGCNA module following a log scale. For each module, a set of genes belonging to the module is indicated on the right side of the heatmap.
Figure adapted from Meistermann et al. *[13].*

e.g., `X`. A string is a sequence of characters enclosed in double quotes, e.g., `"2.34"`. In addition, the token `_` stands for an *anonymous variable*.

Atom An atom is a block comprising two parts: the predicate and the arguments. For example, `birth_place(casper,X)` is an atom with the predicate `birth_place`, and 2 arguments: the constant `capser` and the variable `X`. We define the arity of predicate by the number of arguments linked to the predicate within an atom, denoted as follows: `<predicate>/<arity>`. For instance, `birth_place` will be denoted by `birth_place/3`. Furthermore, a *ground atom* is an atom without any variable. `birth_place(casper,X)` is not a ground atom, while `birth_place(casper,antartic)` is a ground atom.

Rules, facts and constraints An ASP program consists of rules of the form:

$$\underbrace{\mathtt{a_0}}_{\text{head}} \mathtt{:-} \underbrace{\mathtt{a_1}, \ldots, \mathtt{a_m}, \mathtt{not\ a_{m+1}}, \ldots, \mathtt{not\ a_n}.}_{\text{body}}$$

where, $\mathtt{a_1}, \ldots, \mathtt{a_n}$ are atoms, the symbol `:-` can be interpreted as a "if" and the comma (`,`) is a "and". The `not` symbol denotes the (default) *negation*. Intuitively, a rule can be expressed as: "the head is true if the body holds". The particularity of ASP is that the reasoning is *closed world*. In other words, an atom is considered false until proved otherwise. Furthermore, diverse rule adaptions can be made.

A *fact* is a rule without body ($n = 0$) and is always true:

$$\mathtt{p}.$$

Generally, a fact represents knowledge.

A rule without a head ($head = \varnothing$) is a *constraint*:

$$\mathtt{:-\ q}.$$

Therefore what is present in a constraint cannot be present in solutions.

ASP notations ASP allows the use of various notations to enrich programs:

- **Comparison:** Classical mathematical comparisons, such as `=` (equal), `!=` (not equal), `<` (less than), `<=` (less than or equal), `>` (greater than), `>=` (greater than or equal), are available.

- **Condition:** The notation `a(X): b(X)` means that all instances of `a(X)` are inferred for which the corresponding instance `b(X)` appears.
- **Choice rule:** The notation `n {a(X): b(X)} m` is true if at least `n` and at most `m` instances of `a(X)` (subject to `b(X)`) are satisfied. A choice rule can be used in the body (acting as a constraint) or in the head (acting as a rule).
- **Count:** The expression `N = {X: a(X)}` counts the number of distinct `X` for which `a/1` predicate is true.
- **Optimization:** The notation `#maximize{W: a(X)}` (resp. `#minimize{W: a(X)}`) is used to maximize (resp. minimize) the sum of weights `W` subject to `a(X)`.
- **Display:** The command `#show a/1.` displays all true `a/1` predicates.

Further details on ASP notations can be found in the Potassco documentation [2].

Toy example In this section, we illustrate some concepts in a toy example (Program 3.1).

First, consider the problem encoding in lines 3-6. Line 3 defines that a bird can fly by default, unless negated (`not neg_fly(X)`). Here, the default negation takes on its full meaning. Unless explicitly stated that a bird cannot fly using the predicate, it is assumed that the bird can fly. Lines 4 and 5 establish that an eagle and a penguin, respectively, are both birds. Line 6 introduces the `neg_fly/1` predicate, indicating that a penguin cannot fly.

Next, the problem instance is specified in lines 8 and 9, which represent our knowledge base. These lines state that `ricky` is an eagle and `casper` is a penguin. Consequently, `ricky` is inferred to be a bird capable of flight, while `casper` is a bird without the ability to fly.

Finally, lines 11 and 12 restrict the output to display only the `bird/1` and `fly/1` predicates. This selective display ensures that the results focus on the relevant aspects of the problem encoding.

It is important to note that the order of rules and constraints in ASP does not impact the resolution of the program.

2. https://potassco.org

```
1  % This is a comment in ASP
2  % Problem encoding
3  fly(X) :- bird(X), not neg_fly(X).
4  bird(X) :- eagle(X).
5  bird(X) :- penguin(X).
6  neg_fly(X) :- penguin(X).
7  % Problem instance
8  eagle(ricky).
9  penguin(casper).
10 % Display
11 #show bird/1.
12 #show fly/1.
```

Program 3.1 – ASP program toy example.

Upon resolving this program, we obtain the results shown in Figure 3.4. We observe a single solution containing two `bird/1` predicates and one `fly/1` predicate. As expected, `ricky` and `casper` are identified as birds (`bird(ricky)` and `bird(casper)`), with `ricky` being capable of flight (`fly(ricky)`). The program and its results can be accessed and regenerated through the provided notebook link in footnote [3].

```
clingo version 5.5.0
Reading from stdin
Solving...
Answer: 1
bird(casper) bird(ricky) fly(ricky)
SATISFIABLE

Models       : 1
Calls        : 1
Time         : 0.001s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.001s
```

Figure 3.4 – Obtained results for Program 3.1.

3. https://mybinder.org/v2/gh/mathieubolteau/Bolteau_PhD_Thesis_Supplement/master - see Notebook `Illustration_3.1.ipynb`. The web page may take a few minutes to load.

3.3.3 Solving process

ASP allows programmers to solve problems by returning one or more solutions that satisfy the problem's description. The computation process in ASP involves two key stages: grounding and solving (Figure 3.5) [82, 83].

1. **Grounding.** In this initial step, all constraints and rules are instantiated with specific data (facts). This process eliminates variables within the rules and constraints, resulting in a grounded program, comprising only ground atoms. A dedicated program called a *grounder* generates a finite propositional representation of the original ASP program. This essentially translates the program into a form that can be efficiently solved by specialized algorithms.

2. **Solving.** Following grounding, a solver is employed to generate stable models given a grounded program. These stable models satisfy all program rules and optimizations.

Ultimately, the solutions are presented to users, each representing a possible interpretation of the problem based on the stable models.

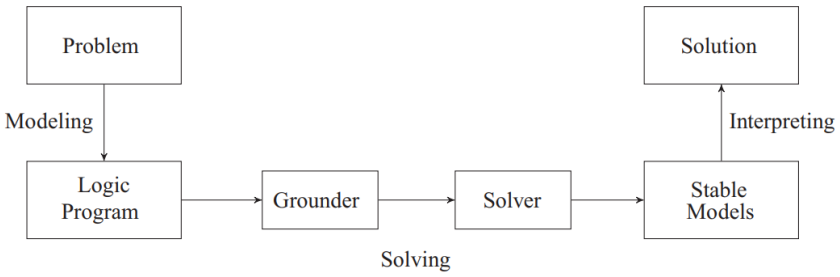


Figure 3.5 – ASP problem resolution process.
A problem is modeled in the syntax of logic programs. Then, ASP process is decomposed in two steps. First, a grounder generates a finite propositional representation of the input program. Secondly, the solver computes the stable models. Finally, the solution is given.
Figure from Gebser et al. *[84].*

Several grounders and solvers exist [85], such as Lparse [86], GIDL [87] and SModels [88], ASSAT [89], respectively. In this thesis, the Potsdam Answer Set Solving Collection (Potassco) and more especially, the grounder Gringo [90] and the solver Clasp [91] are used.

3.3.4 ASP program example

In this section, we present an ASP program example introducing possible representation of cells and gene expression in ASP (Program 3.2). The objective of this program is to select potential perturbations for two classes.

```
1  cell(c1). % c1 is a cell
2  cell(c2).
3  cell(c3).
4  class(early_TE). % early_TE is a class
5  class(medium_TE).
6  class(late_TE).
7  be_part(c1,early_TE). % c1 belongs to early_TE class
8  be_part(c2,medium_TE).
9  be_part(c3,late_TE).
10 gene(g1). % g1 is a gene
11 gene(g2).
12 expr(c1,g1,0). % g1 gene is expressed at 0 in c1 cell
13 expr(c1,g2,0).
14 expr(c2,g1,0).
15 expr(c2,g2,1).
16 expr(c3,g1,1).
17 expr(c3,g2,1).
18 pert(C,G,S,CL) :- expr(C,G,S), cell(C), gene(G), be_part(C,CL).
19 {sel_pert(C,G,S,CL) : pert(C,G,S,CL)}.
20 :- sel_pert(_,_,_,early_TE).
21 #maximize{1,C,G: sel_pert(C,G,_,_), pert(C,G,_,_)}.
22 #show sel_pert/4.
```

Program 3.2 – ASP program of pseudo-perturbation generation.

The ASP program delineates knowledge through facts (lines 1-17). For instance, the `expr(c1,g1,0)` predicate on line 12 signifies that in cell `c1`, the gene `g1` exhibits an expression value of `0`. Line 18 introduces a rule defining a `pert/4` predicate, modeling experimental data, where gene G demonstrates an expression value S in cell C, associated with class CL. Line 19 selects a subset of `pert/4` predicates using `sel_pert/4`,

representing the chosen perturbations. This construct, known as *choice rule*, is central in ASP modeling, generating potential combinations of candidate solutions. These solutions are typically filtered using constraints. For instance, line 20 restricts the solution candidate `sel_pert/4` from being linked with the *early_TE* class. These solutions of this program are rendered as a set of `sel_pert/4` predicates. Each solution validates all program rules. At this point, respecting the rules and constraints of lines $1-20$, 16 solutions are possible, representing all subsets for 2 cells associated with 2 genes. Mathematically, we have $(2^2)^2 = 16$ solutions. The first part (2^2) represents the two cells, c2 and c3, that can be present or absent in the solution, resulting in 4 solutions:

- No cells selected: `{}` (empty set, $\varnothing$).
- Cell c2 only selected: `{sel_pert(c2,G,_,medium_TE)}`.
- Cell c3 only selected: `{sel_pert(c3,G,_,late_TE)}`.
- Both cells c2 and c3 selected: `{sel_pert(c2,G,_,medium_TE)}, {sel_pert(c3, G,_,late_TE)}` .

Furthermore, each `sel_pert/4` predicate involves gene G. Since two genes, `g1` and `g2`, are possible, we add a second exponent $(^2)$ to the solution count. This brings the total number of solutions to $4^2 = 16$. However, our goal is to identify all possible perturbations for the cells present in medium and late TE. To achieve this, we aim to maximize the number of possible perturbations. Line 21 searches to maximize the `sel_pert/4` predicate with the association of cell C and gene G. Behind this `#maximize` statement, a counting of predicates, corresponding to associations, is made by the solver. Finally, line 22 presents the display of the solution(s) of this program, focusing solely on the `sel_pert/4` predicates. Thus, we obtain as a result one solution that maximizes the number of `sel_pert` predicates having different constant terms : `{sel_pert(c2,g1,0,medium_TE), sel_pert(c3,g1,1,late_TE), sel_pert(c2,g2,1,medium_TE), sel_pert(c3,g2,1,late_TE)}`.

To illustrate our purpose, we invite the reader to run this ASP program following the link in foonote[4]. In the notebook, run the first cell (corresponding to the first 21 lines) to obtain the 16 solutions. By running the second cell, the optimization will be performed to obtain the maximal number of `selpert/4` predicates. In Figure 3.6, we present a screenshot of the second cell and its output. The solver

4. https://mybinder.org/v2/gh/mathieubolteau/Bolteau_PhD_Thesis_Supplement/master - see Notebook `Illustration_3.2.ipynb`. The web page may take a few minutes to load.

gives `OPTIMUM FOUND` meaning that an optimal solution is found. This solution is the last found, here: `sel_pert(c2,g1,0,medium_TE) sel_pert(c3,g1,1,late_TE) sel_pert(c2,g2,1,medium_TE) sel_pert(c3,g2,1,late_TE)`. This solution is the one comprising the maximal number of `sel_pert/4` predicates.

[2]:
```
%%clingo  0
cell(c1).
cell(c2).
cell(c3).
class(early_TE).
class(medium_TE).
class(late_TE).
be_part(c1,early_TE).
be_part(c2,medium_TE).
be_part(c3,late_TE).
gene(g1).
gene(g2).
expr(c1,g1,0).
expr(c1,g2,0).
expr(c2,g1,0).
expr(c2,g2,1).
expr(c3,g1,1).
expr(c3,g2,1).
pert(C,G,S,CL) :- expr(C,G,S), cell(C), gene(G), be_part(C,CL).
{sel_pert(C,G,S,CL) : pert(C,G,S,CL)}.
:- sel_pert(_,_,_,early_TE).
#maximize{1,C,G: sel_pert(C,G,_,_), pert(C,G,_,_)}.
#show sel_pert/4.
```

```
clingo version 5.5.0
Reading from stdin
Solving...
Answer: 1

Optimization: 0
Answer: 2
sel_pert(c3,g1,1,late_TE)
Optimization: -1
Answer: 3
sel_pert(c3,g1,1,late_TE) sel_pert(c2,g2,1,medium_TE)
Optimization: -2
Answer: 4
sel_pert(c3,g1,1,late_TE) sel_pert(c2,g2,1,medium_TE) sel_pert(c3,g2,1,late_TE)
Optimization: -3
Answer: 5
sel_pert(c2,g1,0,medium_TE) sel_pert(c3,g1,1,late_TE) sel_pert(c2,g2,1,medium_TE) sel_pert(c3,g2,1,late_TE)
Optimization: -4
OPTIMUM FOUND

Models       : 5
  Optimum    : yes
Optimization : -4
Calls        : 1
Time         : 0.002s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.002s
```

Figure 3.6 – Screenshot of the Program 3.2 and its execution.

The Program 3.2 can be retrieved and run using the following link: https://mybinder.org/v2/gh/mathieubolteau/Bolteau_PhD_Thesis_Supplement/master. To regenerate this result, please `Run` the second cell of the notebook `Illustration_3.2.ipynb`.

3.4 Tools used in this thesis

3.4.1 pyBRAvo: constructing interaction graphs from public databases

pyBRAvo (python Biological netwoRk Assembly) is a computational framework designed to reconstruct human multi-source and multi-pathway gene regulatory networks (GRNs) and signalling networks (SNs) [92]. This tool addresses three primary challenges: *(i)* integrating multiple biological data sources, *(ii)* automating network reconstruction and *(iii)* leveraging biological semantic models to represent causal biological flows.

3.4.1.1 Pathway Commons

pyBRAvo utilizes Pathway Commons (PC) [93], a large collection that aggregates publicly available biological pathway and molecular interaction databases. PC integrates data from 22 databases, such as Kyoto Encyclopedia of Genes and Genomes (KEGG) Pathway [94], Reactome [95], Pathway Interaction Database (PID) [96], among others. This comprehensive integration encompasses over $5,000$ pathways and more than 2 million interactions. PC organizes biological data from these databases into a knowledge graph using linked data principles. Leveraging the semantic web, particularly the SPARQL query language, allows efficient querying of this extensive graph to retrieve specific information.

3.4.1.2 Network reconstruction

pyBRAvo follows the algorithm outlined in Figure 3.7 to reconstruct a network. Initially, the tool takes a list of genes or proteins as input. This list undergoes three preprocessing steps to decompose complexes, add synonyms, and expand suffixes found in databases. Subsequently, pyBRAvo generates a query to send to PC instructing it to identify predecessors directly linked in biological databases to entities in the provided list. The generated query is executed on PC, and the resulting entities are added to the list. This process iterates until no new controllers are found (i.e., no predecessors are identified), or the maximum exploration depth has been reached. pyBRAvo outputs an interaction graph suitable for computational modeling. The provenance database of each entity present in the network is also outputted by the software. Additionally, a powerful advantage of this tool is the reconstruction of an oriented and signed graph. In others words, reconstructed networks reflect the direction of the interaction between entities, as

well as the type of interaction, which can be activation, inhibition, or participation in a protein complex.

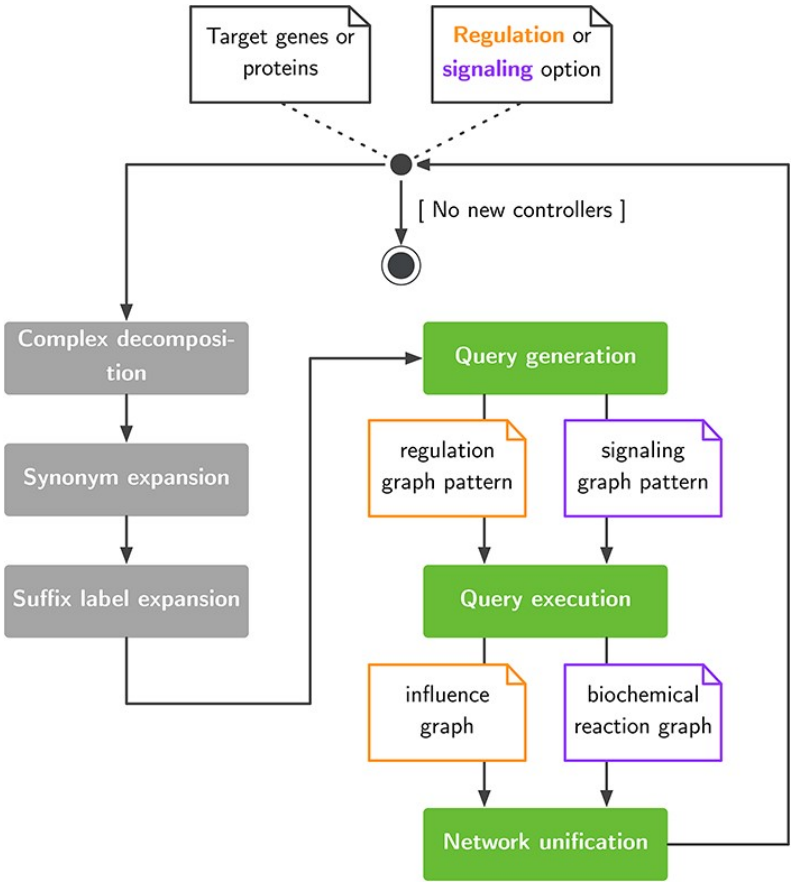


Figure 3.7 – Network reconstruction algorithm of pyBRAvo.
Given a list of genes or proteins, the entities are expanded through three optional steps. A query is then sent to Pathway Commons to identify predecessors of the genes or proteins of the list. The newly identified entities are added to the list and the graph is reconstructed. The querying process continues until no more predecessors are found or the maximum exploration depth is reached. Finally, pyBRAvo generates the reconstructed interaction graph.
Figure from Lefebvre et al. *[92].*

3.4.1.3 Parameters of pyBRAvo

As explained previously, pyBRAvo takes as input a list of entities, genes or proteins, depending on the desired output network (GRN or SN). The tool enables querying regulatory interactions and signaling pathways, which are the primary parameters of the software. The preprocessing steps outlined in the previous section are optional and can

be called using appropriate parameters. Additionally, users have the option to exclude certain databases if deemed irrelevant. Finally, users can set a maximum exploration depth in order to stop the recursive querying process. This last parameter is crucial for determining the depth of prior knowledge we aim to extract: the greater the depth, the more extensive the mechanisms that will be uncovered.

3.4.2 Caspo: learning Boolean models

3.4.2.1 Caspo's workflow

Caspo (Cell ASP Optimizer) [97–99] is an open-source software to learn Boolean networks modeling signal transduction of a biological system. This Python package is designed to construct ASP programs and leverage an ASP solver to execute and find solutions for these programs. The objective of this software is to infer logical models allowing the identification of mechanisms underlying signal transduction characteristics and generating reliable hypotheses. Three main applications, shown in Figure 3.8, can be addressed by Caspo: *(i)* learning BNs, *(ii)* classify these BNs and *(iii)* design new experiments to better discriminate the learned BNs.

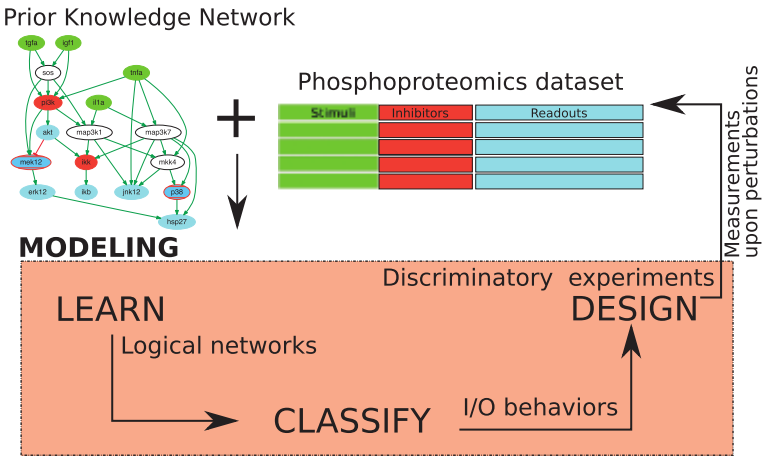


Figure 3.8 – Workflow of Caspo software.
The workflow consists of a loop made of three main modeling steps: *(i)* learn a family of logical networks from a prior knowledge network and a phosphoproteomics dataset; *(ii)* classify networks w.r.t. to their input-output (I-O) behaviors; and *(iii)* design new experiments to discriminate all I-O behaviors.
Figure adapted from Videla et al. *[99].*

3.4.2.2 Inputs of the method

Prior knowledge network, a signaling network The foundation of the method lies in prior knowledge, which comprises signal transduction described as a set of causal interactions between proteins. This information represents the biological knowledge about the studied system modeled and can be represented as a signed and directed graph, called the Prior Knowledge Network (PKN). In this graph, three types of nodes are distinguished:

1. *stimuli* (green nodes): proteins that can be stimulated;
2. *inhibitors* (red nodes): proteins that can be inhibited;
3. *readouts* (blue nodes): output proteins where activity is observed after perturbations of the two previous entities.

Experimental design composed of phosphoproteomics data To better understand the studied problem, diverse phosphoproteomics experiments are conducted. The system is composed of proteins that can be stimulated and/or inhibited. Scientists explore the repercussions of these perturbations on the system through the activity measurement of proteins called readouts [100]. This constitutes what we refer to as an *experimental design*, composed of *(i)* perturbations, i.e. stimulation or inhibition of certain proteins, and *(ii)* readout protein observations. An experimental design can be represented as shown in Figure 3.9, containing n experimental conditions where different combinations of stimuli and/or inhibitions are tested, and their impact on m readouts is observed. Perturbations are represented by a 1 for stimulus and a 0 for inhibition, thus considering perturbations as binarized values. Readout measurements, $\theta_{i,j} \in [0, 1]$, denote the observed activity of a protein j under the experimental condition i, where $0 \leq i \leq n$ and $0 \leq j \leq m$. These observations consist of normalized values between $[0, 1]$.

						m	
	Experiments	Stimuli		Inhibitors		Readouts		
		protein A	protein B	protein C	protein D	protein E	protein F	protein G
n	#1	0	1	0	1	0.4	0.1	0.9
	#2	1	0	1	0	0.2	0.1	0.6
	#3	1	1	0	1	0.3	0.5	0.2

Figure 3.9 – Example of an experimental design.
Three experiments are observed, each comprising 2 stimuli, 2 inhibitors and 3 readouts. Perturbations (stimuli and inhibitors) are binarized values whereas readouts are normalized values between $[0, 1]$.

3.4.2.3 Modeling steps

Learning Boolean networks Given a PKN, representing all possible interactions between proteins, and an experimental design, containing observation data, the objective is to learn Boolean networks (BNs) consistent with the possible interactions and biological observations [98, 99]. For example, it might be obvious that two nodes in the PKN have positive effect on another node, but the graph alone cannot specify whether the target node is active in the presence of either node or only when both are present. BNs that fit observations and prior knowledge can be learned using Caspo, implemented in ASP. Caspo defines rules and constraints to obtain an optimal BN comprising Boolean functions. Given a learned BN denoted by B, for each experimental condition i, we can compute the Boolean prediction $\rho_{i,j} \in \{0, 1\}$ of the state of protein j by using the logic formulas described by B [97]. In addition, we define the size of a BN as the sum of the number of logic gates of each node in the BN [97].

Learning the optimal BN involves two optimizations:

1. Minimization of the mean square error (MSE) to learn BNs with binarized predicted values as close as possible to normalized observations regarding the readouts. The minimization follows the Equation 3.1.
2. Minimization of the size of BN to explain the observations as simple as possible. The size is determined by counting the number of logic gates arriving or regulating the nodes within the BN.

$$MSE = \frac{1}{n \times m} \sum_{i=1}^{n} \sum_{j=1}^{m} (\rho_{ij} - \theta_{ij})^2, \tag{3.1}$$

where n is the number of experiments, $\rho_{ij} \in \rho$, with ρ is the set of predicted readout values, and $\theta_{ij} \in \theta$, with θ is the set of observed readout values.

Beyond the single optimal BN identified, knowledge about the system can be further enriched by leveraging suboptimal BNs. This is achieved by relaxing the two optimization criteria. First, a tolerance level can be introduced for the Mean Squared Error (MSE). This allows us to explore BNs with MSE values ranging from the optimal value up to a user-defined maximum tolerance added to the optimal MSE. We call this criterion the *fitness_tolerance*. Second, a tolerance level can also be introduced for the size of the BN. This allows us to explore BNs with a number of nodes ranging from the optimal size up to a user-defined maximum tolerance added to the optimal number of nodes. This criterion

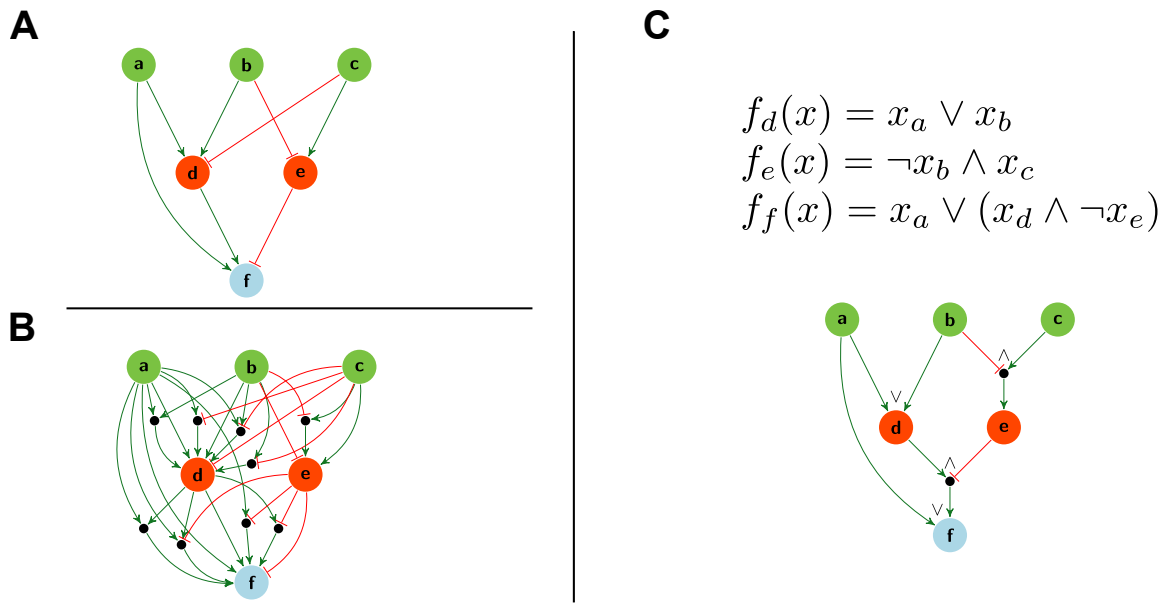
is called *size_tolerance*. Additionally, a third parameter could be considered in the BN learning: the *length*. This value restricts the number of incoming interactions in an "AND" logical gate. Ultimately, all these learned optimal and suboptimal BNs, considering the parameter values, are combined to form a *BN family*, which collectively provides a more comprehensive model of the system.

Let us illustrate the concepts with the toy example in Figure 3.10. Consider a PKN (Figure 3.10A) comprising 6 nodes (with 3 inputs, 2 intermediates and 1 readout), and 8 edges. The hypergraph in Figure 3.10B, represents all plausible logical interactions for the previous PKN. From this hypergraph, multiple BNs are possible to construct. In Figure 3.10C, we present an arbitrary BN, derived from the PKN (i.e., pruning the hypergraph), with its Boolean functions and its directed hypergraph representation.

Given a toy BN, illustrated in Figure 3.10C, and an experimental design composed of two experiments (left table in Figure 3.10D), the expression value of the readout f can be predicted. For instance, with $a = 1$, $b = 0$, $c = 1$, $d = 1$ and $e = 1$ (experiment #1), f is predicted to be 1. We can compare this prediction, with the observed value and thus compute the MSE. All experiments of this toy example lead to a computed MSE equal to 0.03365.

Classifying Boolean networks into input-output behaviors We can classify members of a BN family with respect to their input-output predictions. Briefly, some BNs can be regrouped together because they cannot be distinguished regarding their input and output. In other words, these networks generate the same output (readout prediction) for every possible Boolean value of input (stimuli and inhibitor), i.e., they have the same input-output behavior [101]. Generally, the number of input-output (I-O) behaviors is significantly less than the number of BNs, facilitating the analyses [97].

Design new discriminatory experiments BNs having the same I-O behavior cannot be discriminated based on the available experimental design. However, Caspo provides a functionality in order to design new experimental perturbations to distinguish BNs in a same I-O behavior [101]. Given these new stimuli and inhibitors, wet lab experimental could be lead to supply initial experimental observations and thus improve the learning step. For more details on this subject, we refer the reader to Videla *et al.* [99].



experiment	stimuli			inhibitors		readouts (θ)
	a	b	c	d	e	f
#1	1	0	1	1	1	0.88
#2	0	0	1	0	1	0.23

prediction (ρ)
f
1
0

$$MSE = \frac{1}{n \times m} \sum_{i=1}^{n} \sum_{j=1}^{m} (\rho_{ij} - \theta_{ij})^2$$
$$= \frac{1}{2} \times ((1 - 0.88)^2 + (0 - 0.23)^2)$$
$$= 0.03365$$

Figure 3.10 – Learning Boolean network illustration.
The green nodes represent the protein that can be experimentally stimulated. The orange nodes represent species that can be inhibited. Blue nodes represent the readouts, i.e., the protein that can be measured. Green and red arrows represent activations and in inhibitions respectively.
(A) Example of an interaction graph describing an imaginary prior knowledge network (PKN).
(B) Hypergraph expansion describing all plausible logical interactions for the PKN in (A).
(C) An arbitrary Boolean network (BN) B derived from the PKN in (A) (i.e. pruning the hypergraph in (B)). On the top, are the logical functions of B. On the bottom, is the hypergraph representation of B.
(D) Computation of the MSE given a toy experimental design (left table) and the BN B in (C). Through the BN B, the readout f is predicted to be 1 and 0, respectively, for the two experiments. The calculated MSE is given by the equation on the right.

3.4.3 Using Caspo for modeling a response to a treatment

In this section, we introduce an original study which exploit Caspo to learn BNs and discriminate the response of patients to a treatment. The implemented method served as inspiration for the development of our approach in this thesis.

3.4.3.1 Context of the study

The paper by Chebouba *et al.* [11] presents a method to discriminate the response of acute myeloid leukemia (AML) patients to treatment. AML is a cancer affecting blood cells, where only a quarter of patients survive beyond 5 years. The authors propose studying proteomics data in order to classify the response of the treatment in two classes: complete remission (CR) or primary resistant (PR). They suggest to use prior information signaling networks from publicly available resources, combining to protein levels of 191 patients (from CR and PR classes). The main objective is to derive (learn) pseudo-perturbations for this type of patient dataset where real perturbations cannot be captured, to apply Caspo for learning BNs specific to each patient class.

3.4.3.2 Workflow of the implemented method

The implemented method consists of 4 main steps. First, they reconstructed the prior knowledge network (PKN) from public databases. Then, they used ASP to select proteins and patients. The third step involved inferring Boolean networks (BNs) using Caspo [99]. A BN is inferred for each class, modeling it. Finally, they classify patients into one of the two classes (CR and PR) based on the previously learned BNs. The general workflow of the method is illustrated in Figure 3.11. More details about each step are provided in the following sections.

Create the prior knowledge network The proteomics data consist of measurements of 191 AML patients, with 231 protein levels measured. These patients are classified into two classes: complete remission (CR) for those who had a good response to the treatment, and primary resistant (PR) for those who did not respond well. To reconstruct the PKN, the authors utilize the KEGG database [10], through a plugin that queries the database and constructs the network (Figure 3.11a). Each node of the network represents a gene, and edges represent interaction between these genes. They define three types of nodes: stimuli, which are nodes without predecessors, readouts, which are nodes without successor, and inhibitors, which are the other genes.

Selection of proteins and patients To use the Caspo framework, the proteomics data need to be divided into entry and output measurements. The entry comprises a list of perturbations containing stimulus and inhibitor information, while output consists of

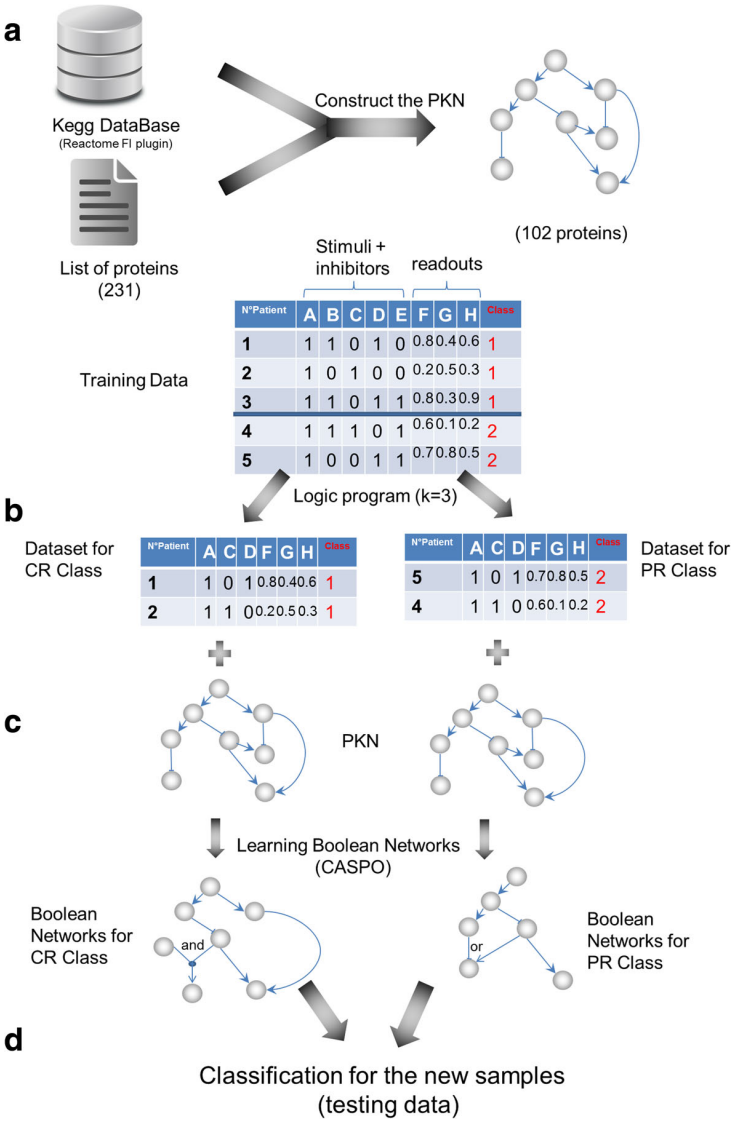


Figure 3.11 – Workflow of the method to discriminate the response to a treatment.

(a) PKN reconstruction. Using the proteins present in the dataset as input, KEGG database where queried to construct the PKN.
(b) Protein and patient selection. k proteins are selected from the dataset based on pairs of patients with identical values in these proteins but belonging to different response classes.
(c) Boolean network (BN) learning. BNs are generated for response classes CR-PR using datasets from step (b).
(d) Classification. Unknown patient datasets are classified using the learned logic models.
Figure from Chebouba et al. *[11].*

the readout observations. Recall that the objective of Caspo is to learn BNs answering to the entry-output relation.

The authors preprocess proteomic data by discretizing the proteins corresponding in the PKN to stimuli and inhibitors, while readout protein are normalized between $[0, 1]$ (Figure 3.11b). They design an ASP program in order to select proteins and patients according to specific rules:

1. Select k proteins from all possible combinations of stimuli and inhibitors.
2. Select pairs of patients from each class (CR and PR) for which the k selected proteins have the same values.
3. Maximize the number of pairs of patients.
4. Maximize the difference of readout proteins of the selected pairs of patients.

The ASP program allows constructing two experimental designs, each specific to a class, where they share the same entry (stimuli and inhibitors), but have a different output (readout values). For more details about this method, readers can refer to Chebouba *et al.* [11].

Boolean networks learning Combining the previously reconstructed PKN and the experimental designs, the authors use Caspo to learn BNs (Figure 3.11c). One BN family is inferred for each class (CR and PR). These two BN families represent logic models composed of signaling interactions, specific to each class.

Classification of patients Finally, they suggest a classification step to predict the response to treatment for new patients using the previously learned BNs (Figure 3.11d). Given a dataset from a new patient, one can predict the readout protein value through the two families of learned BNs. They calculate the difference between these predictions from BNs and the readout observations using the MSE. Then, they classify the new patient to the class with the lowest MSE.

This chapter has provided the necessary background for the thesis. The tools introduced here will be employed in the methodology implemented in the subsequent chapter. Furthermore, ASP will play a crucial role in our methodology for identifying pseudo-perturbations from single-cell transcriptomic data.

Chapter 4

Contribution 1: Inferring Boolean networks to model human preimplantation development

"All models are wrong, but some are useful."

— George E. P. Box (1919-2013)

"Models should be as simple as possible, but not simpler."

— Albert Einstein (1879-1955)

Summary

This chapter outlines the framework I developed, which involves inferring Boolean networks from both prior knowledge and single-cell transcriptomic (scRNAseq) data. The work presented here brings together key findings from two scientific articles, offering a unified perspective. The first, Bolteau *et al.* [5] was published in the proceedings of the 19th International Symposium on Bioinformatics Research and Applications (ISBRA 2023). I also presented our work at the conference in Wrocław, Poland. The second, Bolteau *et al.* [6], published in the Journal of Computational Biology (JCB) in 2024, provides a deeper exploration of the first paper's findings. Our methodology primarily focuses on inferring computational models capable of distinguishing between two developmental stages. Our method selects pseudo-perturbations from scRNAseq data since actual perturbations are impractical due to ethical and legal constraints. By combining these pseudo-perturbations with prior regulatory network, we are able to infer Boolean networks that accurately align with scRNAseq data for each developmental stage. Furthermore, I present the results obtained from applying this method to the study of

trophectoderm (TE) maturation. As a result, we infer families of Boolean networks corresponding to both medium and late TE developmental stages. The structural differences between these networks unveil contrasting regulatory pathways, offering valuable biological insights and hypotheses within this domain.

4.1 Introduction

In this contribution, we propose a framework to discover a family of Boolean networks (BNs) of human preimplantation development that captures the discrepancy from one developmental stage to another one. This framework uses a prior knowledge

network (PKN) as a base on which the single-cell transcriptomic (scRNAseq) data is mapped. Then, it identifies pseudo-perturbations specific for two developmental stages. These pseudo-perturbations are used in the last step to infer stage-specific BN models. Perturbation data is useful information to infer BN models [97, 99]. For this case study, perturbation data is rarely available due to practical and legal concerns, our main contribution was to extract pseudo-perturbation data from scRNAseq data, considering its high redundancy. We used the Pathway Commons database [93] to build a PKN and discovered 20 pseudo-perturbations (across 10 genes) characterizing medium and late stages of trophectoderm (TE) maturation. They correspond to the gene expression of 20 cells in each stage; representative on average of 82% of the total cells. Pseudo-perturbations referring to 10 (entry) genes expression were connected (PKN information) to 14 genes (output) expression. The 20 entry-output gene expression configurations allowed us to infer 2 families of BNs (composed of 8 and 15 logic gates) characterizing medium and late TE developmental stages.

4.2 Method

4.2.1 General presentation

Our method involves three steps aimed at constructing stage-specific Boolean networks (BNs), as depicted in Figure 4.1. Initially, we reconstruct a prior knowledge network (PKN) by querying a biological knowledge database to establish gene-gene interactions. Subsequently, we establish an experimental design tailored to each developmental stage, outlining the entries and outputs essential for BN inference. Finally, we integrate the PKN with the previously created experimental designs to derive stage-specific BNs, using Caspo software. Further elaboration on these steps is provided in the following sections. Our implemented framework, along with the complete set of data and results, can be accessed publicly at the following link: https://doi.org/10.5281/zenodo.10580801.

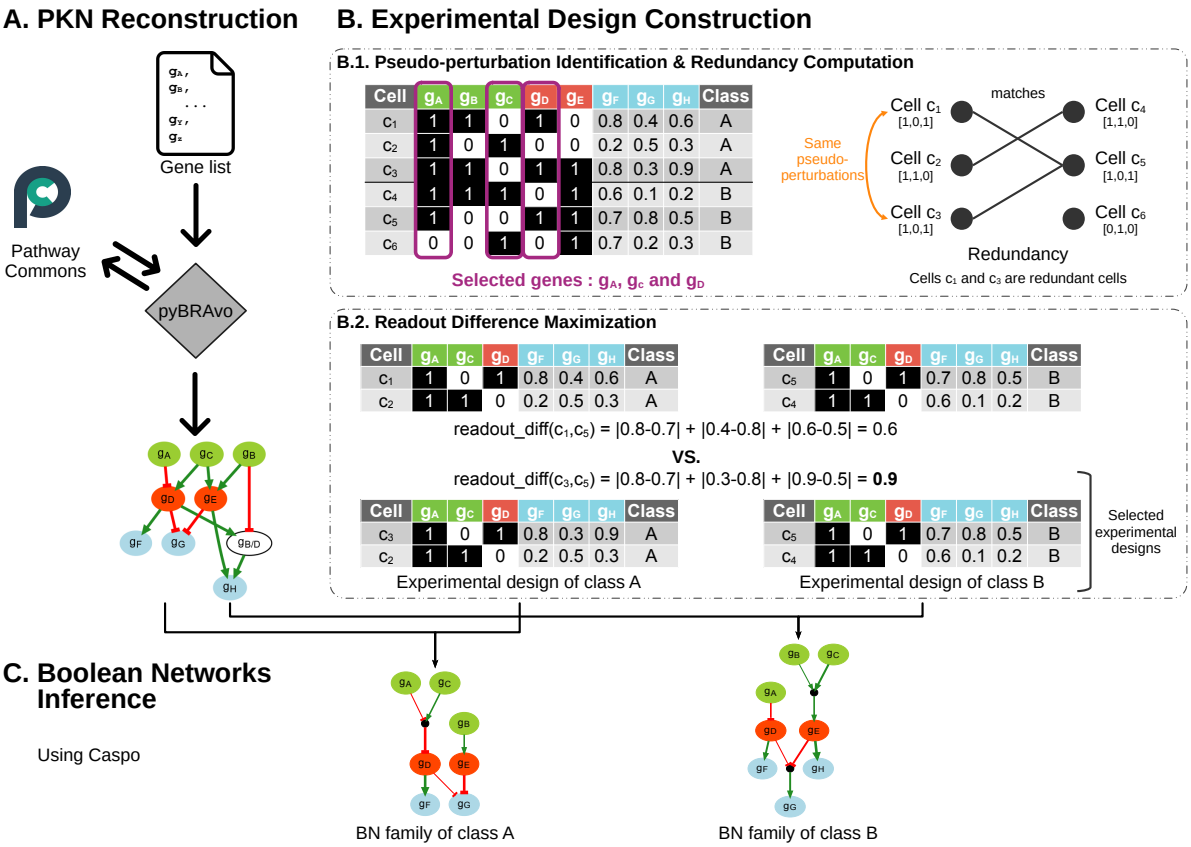


Figure 4.1 – Developed framework comprising three main steps.

(A) The PKN was reconstructed using the pyBRAvo tool that queries the Pathway Commons database with an input gene list.

(B) The ASP program selects a set of k genes (here, $k = 3$ with g_A, g_C and g_D as selected genes) to maximize pseudo-perturbation identification. In this example, 2 (optimal) pairs of pseudo-perturbations are identified: (c_1, c_5) and (c_2, c_4). Redundancies in scRNAseq data are observed, with c_3 sharing the same Boolean vector as c_1, representing two equivalent solutions. The second sub-step identifies pseudo-perturbations maximizing readout difference between the two classes, leading to the selection of pairs (c_3, c_5) and (c_2, c_4), forming the experimental design.

(C) Boolean networks (BNs) are inferred using the Caspo tool, combining the reconstructed PKN and both experimental designs. Each BN is compatible with the PKN topology and minimizes the gene expression error in the (entry-output) experimental designs.

4.2.2 Definition of pseudo-perturbation

As previously seen (see Section 3.4.2), Caspo needs perturbations to infer BNs. However, in the context of human embryonic development, due to diverse factors such as ethical or legal concerns, perturbing the system is impossible. That is why we introduced the concept of *pseudo-perturbation*.

Definition 4.1 Pseudo-perturbation. *A pseudo-perturbation represents a Boolean vector that encodes the expression status of a specific set of k genes within a particular cell. In the context of comparing cells across various classes or developmental stages, a match refers to a pair of* 2 *pseudo-perturbations from different classes that exhibit an identical gene expression vector, signifying similarity in genetic activity.*

We illustrate, in Figure 4.2, this concept with a toy example of 2 pseudo-perturbations for each studied class (A and B) extracted from the Figure 4.1. The pseudo-perturbation of the cell c_1 matches with the one of cell c_5, where we observe the same Boolean vector for genes g_A, g_C and g_D: $[1, 0, 1]$. We observe also a *match* between pseudo-perturbation of cells c_2 and c_4, with the identical vector $[1, 1, 0]$. In this toy example, we have two pairs of cells illustrating two pseudo-perturbation matches.

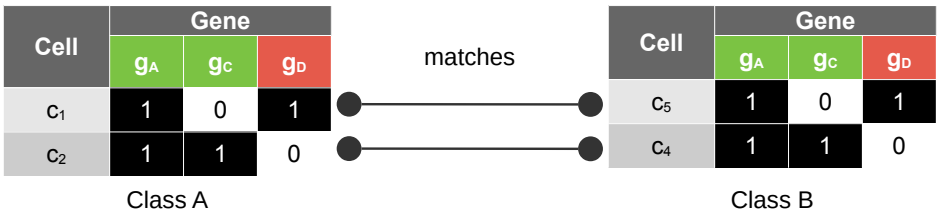


Figure 4.2 – Example of pseudo-perturbations.
Each class (A and B) contains 2 pseudo-perturbations. The two perturbations of a specific class have a match with another perturbation of another class. Here, two pairs of matches can be observed: $\{c_1, c_5\}$ and $\{c_2, c_4\}$.

4.2.3 Data

4.2.3.1 Datasets

As our focus is directed towards investigating the trophectoderm (TE) cell fate, specifically targeting the medium TE and late TE developmental stages, we derive a dataset, called *SC*, from the initial dataset containing the count matrix introduced in Section 3.2. Moreover, we curated several data subsets to test various case studies, as detailed in Table 4.1. We also included an extra case study (P in Table 4.1) from an alternate dataset with phosphoproteomics data sourced from Chebouba *et al.* [11].

Table 4.1 – Datasets description.

Dataset	Source	Class name (C1;C2)	Genes[1]	Cells[2]	#C1's cells[2]	#C2's cells[2]
A	artificial	C1;C2	10	10	5	5
B	subset of single-cell data	E^{TE} ; M^{TE}	30	24	12	12
C	subset of single-cell data	E^{TE} ; M^{TE}	100	50	25	25
D	subset of single-cell data	E^{TE} ; M^{TE}	120	200	100	100
SC	single-cell data	M^{TE} ; L^{TE}	111	680	348	332
P	phosphoproteomics data[3]	CR ; PR	79	191	136	55

E^{TE} = early TE ; M^{TE} = medium TE ; L^{TE} = late TE ; CR = Complete Remission ; PR = Primary Resistant (see [11]). [1] For dataset P, proteins are studied (not genes). [2] For dataset P, patients are studied (not cells). [3] From Chebouba *et al.* [11].

4.2.3.2 Preprocessing

First, we discretize raw gene expression data of input and intermediate PKN nodes (see Section 3.1.1) by considering a gene to be expressed if at least 2 reads are identified in the raw data.

The discretization formula is:

$$e_{ij} = \begin{cases} 0, & \text{if } r_{ij} < 2, \\ 1, & \text{otherwise.} \end{cases} \tag{4.1}$$

Here, e_{ij} is the binarized expression of the gene j for the cell i and r_{ij} is the raw expression

(number of reads) of the gene j for the cell i.

Secondly, we normalize the readout gene expression between 0 and 1. We perform a "min-max" normalization by identifying the minimum and maximum expression measurements of all readout genes across all cells involved in the studied developmental stages.

The normalization formula is:

$$n_{ij} = \frac{r_{ij} - r_{min}}{r_{max} - r_{min}}, \tag{4.2}$$

where n_{ij} is the normalized expression of the gene i for the cell j, r_{ij} is the raw expression of the gene i for the cell j, and r_{min} (resp. r_{max}) is the minimum (resp. maximum) expression value of all readout genes across all cells involved in the studied developmental stages.

4.2.4 PKN reconstruction

4.2.4.1 PKN generation

In our framework, the reconstruction of the PKN relies on the utilization of the pyBRAvo tool [92]. Starting from a list of genes, pyBRAvo employs queries on the Pathway Commons database [93] to identify the predecessors of the initial genes (Figure 4.1A). This iterative process continues until a specified reconstruction depth is reached. For more details, we refer the reader to Section 3.4.1. Through pyBRAvo, we generate a gene interaction graph, serving as the foundational knowledge base for our methodology. Pathway Commons v.13 is leveraged, excluding miRTarBase, MSigDB, and CTD databases to eliminate miRNA and toxicogenomics interactions.

4.2.4.2 PKN reduction

Subsequently, this reconstructed PKN undergoes a reduction process to align with the genes present in the count matrix. We also kept protein-complexes between these genes. Furthermore, inputs directly linked to a single direct successor readout are eliminated to focus solely on gene regulation pathways involving various gene types. Given the topology of the PKN, we determine three types of genes: inputs, intermediates and readouts.

4.2.5 Experimental design construction

Given the reduced PKN and the scRNAseq data of the two studied cell classes, we construct an experimental design for each class (Figure 4.1B).

Definition 4.2 Experimental design. *An experimental design comprises two parts:*

1. *pseudo-perturbations, which are binarized expression values for input and intermediate genes in chosen cells whose value is identical in both cell classes;*
2. *readout observations associated to the cells involved in pseudo-perturbations, which are normalized expression values.*

In the toy example in Figure 4.3A, we observe 2 pseudo-perturbations composed of 3 gene expressions for the class A's experimental design. In addition, the two involved cells match with two other cells in class B, which is also characterized by an experimental design. For both experimental designs, we observe the same expression values for the input and intermediate genes (g_A, g_C and g_D), while different expression values are shown for the readout genes ($g_F - g_H$). Two matching experimental designs can be represented in an alternative way as in Figure 4.3B. On the left, gene expressions for the two experimental designs are depicted, where black (resp. white) rectangle signifies a 1 (resp. 0), i.e., a presence (resp. absence) of the gene. On the right, we show the evolution of the gene expression between the two classes.

To capture the diversity of genes expression in scRNAseq data for each class, we implement an ASP program to maximize the number of different pseudo-perturbations for k genes, given a set of input and intermediate genes (see Section 4.2.5.1). The resulting experimental design is based on the inputs, intermediates, and readouts of the PKN obtained in the previous step.

This experimental design construction algorithm receives an integer k, as a parameter, limiting the number of genes to be selected. Its input data is the preprocessed scRNAseq matrix for input, intermediate, and readout PKN genes. The algorithm retrieves (i) a maximal number of pseudo-perturbations, which identify cells associations between two classes holding identical expression values for a set of k genes, and (ii) cells associations which maximize the readout difference across (redundant) cell associations. The details of this algorithm are presented in the following sections. In addition, we illustrate the experimental design construction process with a toy example in Section 4.2.5.3.

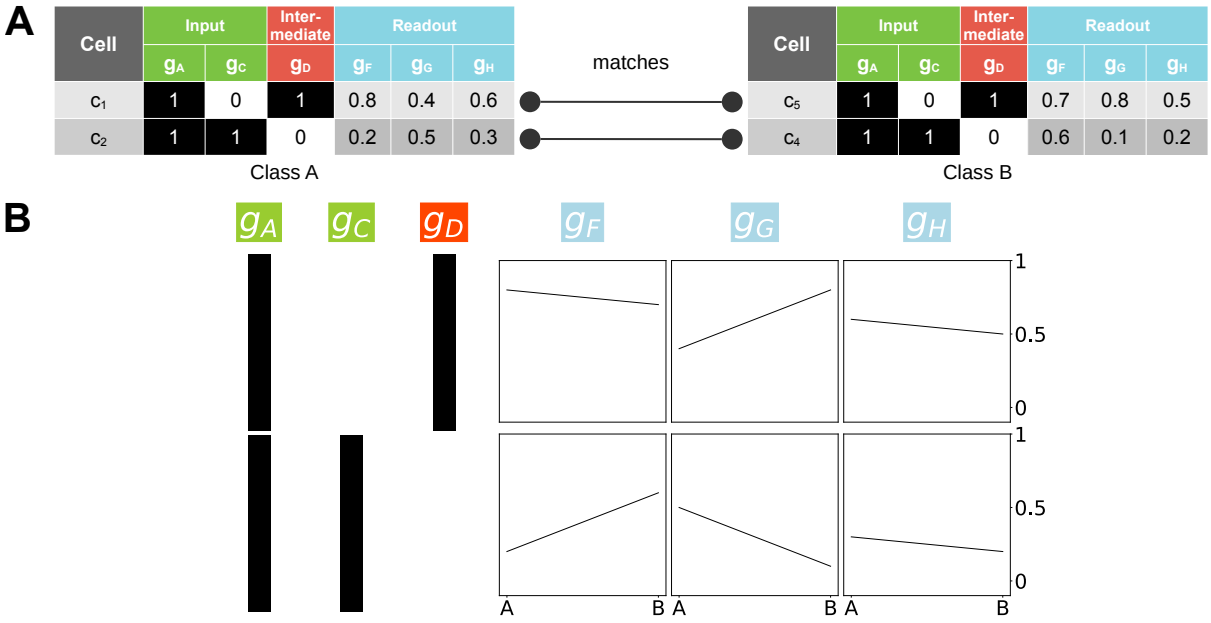


Figure 4.3 – Example of an experimental design.
(A) Two experimental designs comprising 2 pseudo-perturbations, with 2 inputs in green, 1 intermediate in red and 3 readouts in blue. The first, one the left characterizes the class A, while the second on the right characterizes the class B.
(B) Graphical representation of two matching experimental designs. Each row (left side) represents a pseudo-perturbation on the 3 selected input (green) and intermediate (red) genes. Binarized vectors are illustrated using bars, where a black (resp. white) bar means the gene is active (resp. inactive). On the right side, readout genes in blue are shown. In each box, the curve represents the normalized readout gene expression evolution between the class A (A, left) and class B (B, right).

4.2.5.1 Pseudo-perturbation identification

Problem statement The input of this method is a binary matrix, E, where e_{ij} represents the presence or absence of gene j for cell i (see Section 4.2.3.2). The output is a subset of genes and cells that adhere to various constraints, ensuring their pseudo-perturbations are balanced between the two classes.

Let us denote by C, the complete set of cells; and by G, the complete set of genes in our experimental data. Each cell is uniquely associated with one class (either A or B); $C = A \uplus B$. We use the binary matrix, E, to define the relation I^G, $I^G(c_i) = \{g_j \in G | e_{ij} = 1\}$. $I^G(c_i)$ thus represents the active genes, belonging to G, for cell c_i. If $G' \subset G$, then the restriction of I^G to G' is defined by $I^{G'}(c_i) = I^G(c_i) \cap G'$.

Problem formulation. Given an association matrix E, associating a set G of genes to a set C of cells, where C is composed of cells belonging to 2 disjoint sets (classes) A and B; and given a parameter k limiting the number of selected genes, find a subset G' of genes and the largest subset C' ($C' = A' \uplus B' \subset C$, where $A' \subset A$ and $B' \subset B$) satisfying the three following constraints:

Constraint 1. The size of G' is fixed to k (parameter). For large instances $k << |G|$.

Constraint 2. $\forall c_1, c_2 \in A'$ (resp. B'), $c_1 \neq c_2$, $I^{G'}(c_1) \neq I^{G'}(c_2)$.

Constraint 3. $\forall c_1 \in A'$ (resp. B'), $\exists c_2 \in B'$ (resp. A'), such that $I^{G'}(c_1) = I^{G'}(c_2)$.

From this result, for each $c_i \in C'$ we define a binary vector b^i, such that for $j \in \{1, \cdots, k\}$, $b^i_j = 1$ (resp. $b^i_j = 0$) if gene $g_j \in I^{G'}(c_i)$ (resp. $\notin I^{G'}(c_i)$). b^i is called a pseudo-perturbation. Notice that since the sets G' and C' are not unique, there may exist several pseudo-perturbations vectors.

Finally, we optimize $n = |b^i|$, the number of pseudo-perturbations using a maximization: $\max n$. This aims to identify as many pseudo-perturbations as possible to enhance the robustness of the BN inference

Constraints justification. The imposed constraints are crucial in light of the entire framework, which encompasses Boolean network inference and single-cell data.

Constraint 1. This constraint reduces the search space, improves computational efficiency, and simplifies the subsequent step of learning Boolean networks.

Constraint 2. The second constraint prevents redundancy in gene selection from different cells within the same class. This is essential due to the abundance of zero values and redundancy in single-cell data.

Constraint 3. The last constraint promotes similarity in gene expression values between the two distinct classes. This consistency enables meaningful comparative analysis during the subsequent step of Boolean network inference. Despite the inherent evolutionary differences between cells belonging to different classes, selecting genes with similar expression values allows us to impose comparable entry conditions on the system, facilitating accurate modeling of the distinct regulatory mechanisms at play. This *Constraint 3* enables comparable analysis in the subsequent step of Boolean network inference. Finally, selecting a larger number of pseudo-perturbations, using the pseudo-

perturbation maximization, provides more information, enriching the Boolean network inference step and allowing for exploring various regulatory mechanisms.

In addition to this problem formulation, we present in Section 4.3.1 a line by line description of the ASP logic program.

4.2.5.2 Maximization of readout differences

Pseudo-perturbations identified by the previous algorithm relate cells in A' to those in B', forming matching pairs of cells. However, a cell of a specific class can be involved in multiple matching pairs with the same expression vector. To determine which pair to select, we apply a readout difference maximization.

Problem formulation. Given a set of pseudo-perturbation binary vectors, O, and given the matrix of preprocessed scRNAseq data of normalized readout values, find the sets of cells A'^* and B'^*, associated to all pseudo-perturbation vectors in O, that maximize the difference of readout vectors, $r^{A'*}$ (for readouts of cells in A'^*) and $r^{B'*}$ (for readouts of cells in B'^*).

Algorithm. For each vector b in the set of optimal pseudo-perturbations, relating cells c_1 (in A') and c_2 (in B'):

1. Compute a set of *redundant cells* for each class. This involves identifying cells in class A with an identical binarized vector b, denoted as set R_b^A, and likewise for class B denoted as R_b^B. Both sets, R_b^A and R_b^B, include cells c_1 and c_2 respectively.
2. Iterate across all pairs of cells in $R_b^A \times R_b^B$, and calculate the difference of readout gene values while keeping the maximal difference.

We retrieve an association of each optimal pseudo-perturbation to a vector of normalized readouts expression that maximizes the difference between the two classes. Additionally, we calculate the *representativity score* for the optimal pseudo-perturbations by considering the number of redundant cells. Let n^A be the number of cells in class A, and let O be the set of Boolean vectors in all optimal pseudo-perturbations for class A.

The representativity score S^A for class A is defined as follows:

$$S^A = \frac{\sum_{b \in O} |R_b^A|}{n^A}. \tag{4.3}$$

4.2.5.3 Illustration of the experimental design construction process with a toy example

In this section, we illustrate the construction of the experimental design through a toy example. We consider a toy count matrix that includes the expression of 3 binarized input genes, 2 binarized intermediates genes, and 3 normalized readout genes for 3 cells in two classes (Figure 4.4A).

Pseudo-perturbation identification The first objective is to maximize the number of identified pseudo-perturbations. In this example, we set $k = 3$, indicating our aim to identify pseudo-perturbations containing the expression of 3 genes. In this scenario, our method selects genes g_A, g_C and g_D in order to maximize the number of pseudo-perturbations (Figure 4.4A). As defined in Definition 4.1, a cell involved in a pseudo-perturbation matches with another cell of a different class, forming a pair. Both cells exhibit identical Boolean vectors (for the k selected genes). Our toy example exhibits two pairs of cells (Figure 4.4B). The first pair consists of cells c_1 and c_5, both displaying the vector $[1, 0, 1]$. Similarly, the second pair comprises cells c_2 and c_4, sharing the vector $[1, 1, 0]$. Thus, each class (A and B) comprises a maximal number of pseudo-perturbations equal to 2. However, redundancies arise, notably with cell c_3 sharing the same Boolean vector as cell c_1, representing two equivalent solutions for maximizing the number of pseudo-perturbations (Figure 4.4B). To resolve this, a maximization of readout differences is undertaken. The underlying concept is to maximize the divergence in output (readout expression) between the two stage-specific experimental designs.

Readout differences maximization We identify 2 equivalent solutions. For class A, we count 2 redundant cells: c_1 and c_3, while class B does not contain any redundancies. In this instance, our objective is to determine which cell to select between c_1 and c_3. We respectively calculate the readout differences between these cells (c_1 and c_3) and c_5 resulting in readout_diff(c_1,c_5) = 0.6 and readout_diff(c_3,c_5) = 0.9 (Figure 4.5). Given that the latter difference is maximal, c_3 is selected. In summary, the maximization

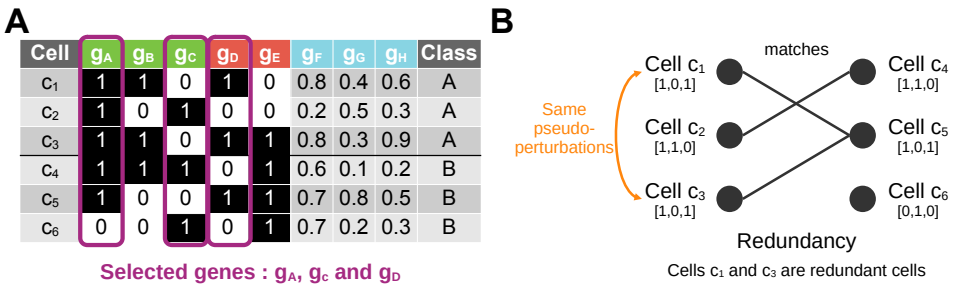


Figure 4.4 – Illustration of the pseudo-perturbation identification.
(A) The method selects a set of $k = 3$ genes, consisting of genes g_A, g_C and g_D. The subsequent expression vectors for the selected genes in each cell are computed.
(B) Two optimal pseudo-perturbations are identified for classes A and B, involving the cells c_1 and c_2, and cells c_4 and c_5, respectively. Redundancies in scRNAseq data are observed, with cell c_3 sharing the same Boolean vector as cell c_1, representing two equivalent solutions.

of readout differences leads to the selection of pairs (c_3, c_5) and (c_2, c_4), forming the experimental designs.

Additionally, for this toy example, we compute the representativity scores (see Equation 4.3) for classes A and B:

$$S^A = \frac{\sum_{b \in O} |R_b^A|}{n^A} = \frac{(2+1)}{3} = 1,$$

where n^A is the number of cells in class A, here 3, and O is the set of Boolean vectors in all optimal pseudo-perturbations for class A, here $|O| = 3$.

$$S^B = \frac{\sum_{b \in O} |R_b^B|}{n^B} = \frac{(1+1)}{3} = 0.667,$$

where $n^B = 3$, and $|O| = 2$.

Thus, for class A, the two pseudo-perturbations represent 100% of the total cells, while the representativity score for class B is 66.7%.

Finally, for this example, we compute two experimental designs specific to each studied class, each comprising 2 pseudo-perturbations. Note that we have the same Boolean expression for input and intermediate genes, but different readout gene expression (Figure 4.5).

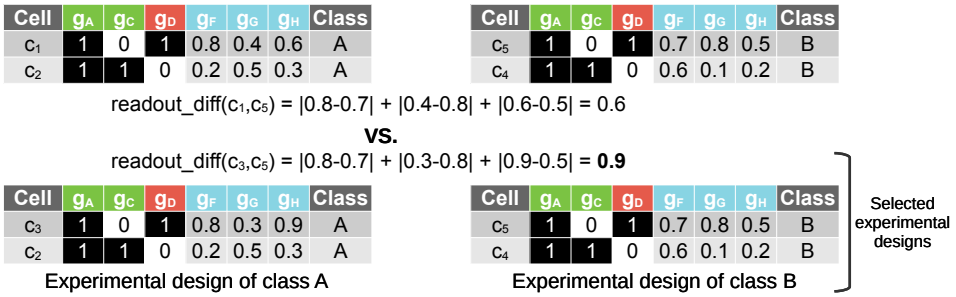


Figure 4.5 – Illustration of the readout difference maximization.
As class A contains redundant cells (c_1 and c_3), the algorithm selects the one that maximizes the readout differences. The `readout_diff()` function is applied to the two potential pairs (c_1, c_5) and (c_3, c_5), yielding a maximal difference for the pair (c_3, c_5). Thus, cell c_3 is selected to be part of the pseudo-perturbations.

4.2.6 BNs inference

Description Here, we outline the process of inferring Boolean Networks (BNs) for each studied class using Caspo [99] (Figure 4.1C). The goal is to derive BNs that accommodate both biological knowledge represented by gene interactions in the PKN and the observed gene expression in the case study. Using logical rules and constraints, Caspo computes BNs that best fit the data through optimizations (using the mean squared error function, MSE), allowing an optional fixed tolerance. The method outputs a set of BNs for each class, derived from the same input (prior knowledge and pseudo-perturbations data) but with different outputs (readout values). These BN families encapsulate knowledge and observations by establishing logical connections between genes. Comparing these BN families enables the identification of distinct behaviors between classes.

Illustration of the BN inference process with a toy example Once again, we illustrate the BN inference step through a toy example (Figure 4.6). Consider a reconstructed Prior Knowledge Network (PKN) comprising 9 genes, with 3 readouts. Additionally, consider class-specific experimental designs consisting of 2 pseudo-perturbations and their readout associated expression value for genes g_F, g_G, and g_H (from the toy example illustration in Section 4.2.5.3).

Employing Caspo [99] and integrating the PKN and the class-specific experimental designs, we infer a BN family for each class. Briefly, we observe distinct regulatory mechanisms modeling the class observations. For instance, in class A, a combination of g_A and g_C (with an "AND" logic gate) is necessary to influence g_D, whereas in class B,

gene g_A alone inhibits gene g_D, while gene g_B is associated with gene g_C to activate gene g_E. Moreover, in class A, only 2 readout genes (g_F and g_G) are considered for modeling, while in class B, all readouts are essential.

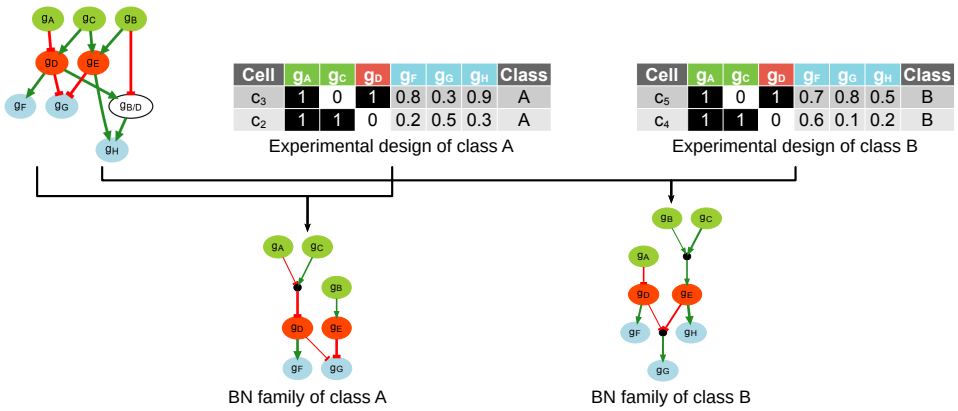


Figure 4.6 – Illustration of the BN inference.
Two executions of Caspo are conducted to learn Boolean network (BN) families, each specific to a class. The same PKN was used, while two different experimental designs were employed (note the differences in readout gene expressions in blue). Different regulatory mechanisms can be observed in the two BN families.

4.3 Results

4.3.1 ASP program

In this section, we provide a step-by-step explanation of the ASP program devised to optimize the count of pseudo-perturbations. Our program, based on the method proposed in Chebouba *et al.* [11], differs primarily in the rule governing the generation of distinct Boolean pseudo-perturbation vectors (see Section 4.2.5.1, Problem statement, *Constraint 2*). Our logic program is tailored specifically to handle scRNAseq data, which often exhibits redundancy due to cells within the same developmental stage sharing identical gene expressions. Additionally, scRNAseq data commonly presents a strong abundance of zero values.

In Program 4.1, we present the ASP program to identify pseudo-perturbations.

```
1  {selgene(G) : pert(C,G,S,CL)} = k.
2  selpert(C,G,S,CL) :- selgene(G), pert(C,G,S,CL).
3  equal(I,J,G) :- selpert(I,G,S1,C1), selpert(J,G,S2,C2), C1<C2, S1=S2.
4  countequal(I,J,M) :- M = {equal(I,J,_)}.
5  0{match(I,J)}1 :- countequal(I,J,k).
6  nbInputOnes(C, N) :- N = {pert(C,G,1,_) : selinput(G)}, match(C,_).
7  :- match(C,_), nbInputOnes(C,N), N<1.
8  diff(I1,I2,G) :- selpert(I1,G,S1,C1), selpert(I2,G,S2,C2), C1=C2,
       S1!=S2, I1<I2.
9  countdiff(I1,I2,M) :- M = {diff(I1,I2,_)}.
10 :- countdiff(I1,I2,0), match(I1,_), match(I2,_), I1<I2.
11 :- countdiff(I1,I2,0), match(_,I1), match(_,I2), I1<I2.
12 #maximize{1,I: match(I,_)}
```

Program 4.1 – ASP encoding of pseudo-perturbation identification.

The `pert/4` predicate, an instance in our program referring to experimental data, emerges from discretized scRNAseq data associated with input and intermediate genes. It delineates the expression of gene G at value S in cell C, linked to class CL. Starting from line 1, our logic program initiates by selecting a set of k genes from all potential input and intermediate genes using the `selgene/1` predicate. This process generates $\binom{m}{k}$ answer sets, where m denotes the total count of input and intermediate genes. The subsequent rules aim to filter these candidate answer sets. Line 2 introduces the `selpert/4` predicate, summarizing experimental data for the selected genes. Following this, line 3 employs the `equal(I,J,G)` predicate to identify pairs of cells I and J from distinct classes (`C1<C2`), exhibiting the same measured value S for gene G (`S1==S2`). Moving forward to line 4, the `countequal(I,J,M)` predicate enumerates the count of genes M demonstrating identical values across cells I and J. Recall that we are interested in finding k identical values associations for k genes. Therefore, in line 5, we define the predicate `match(I,J)`, representing a matching pair of cells I and J having the same expression for the k selected genes. Notice that the terms `match/2`, represented by variables I and J, pertain to cells in the first and second classes, respectively. To discard the redundancies, we generate either 0 or 1 occurrence of the matching pair, to keep only distinct pseudo-perturbations.

Lines 6-7 introduce rules to handle data sparsity. Line 6 defines `nbInputOnes/2`, calculating the count of input genes having a value of `1` for a cell C selected by the `match/2`

predicate. Then, line 7 prohibits the selection of an `match(C,_)` if the count of 1-valued input genes in cell C is less than 1 (`N<1`). This ensures that each pseudo-perturbation has at least one active input gene, disallowing vectors where all genes remain inactive (equal to 0).

To discern distinct pseudo-perturbations within the same class, additional rules (lines 8-11) are introduced. Line 8 defines the `diff(I1,I2,G)` predicate, selecting cells $I1$ and $I2$ from the same class but with distinct values for gene G (`S1!=S2`). The subsequent `countdiff/3` predicate on line 9 records the disparities in expression values of the selected genes between cells $I1$ and $I2$. In line 10 (resp. line 11), the constraint forbids predicates `countdiff(I1,I2,0)`, where there is no difference in expression values for the selected genes, for cells $I1$ and $I2$ selected to be affinities in line 5 for the first class (resp. for the second class). Combined lines 5, 10, and 11 keep only one matching pair among all redundancies in the output solution. For instance, in the first class illustrated in Figure 4.7A, we retain only one match between `match(c1,c3)` and `match(c2,c3)`. Similarly, for the second class, we select either match(c4,c5) or `match(c4,c6)` (Figure 4.7B). In these cases, the selected matching pair will be inferred by the line 5, while the other will not.

Figure 4.7 – Illustration of handling cell redundancies in the ASP program.
(A) Cells $c1$ and $c2$ are redundant in the first class because they match with $c3$. Combining the lines 5 and 11 of the Program 4.1, the ASP program will select only one `match/2` predicate, i.e. either `match(c1,c3)` or `match(c2,c3)`.
(B) Similarly, cells $c5$ and $c6$ are redundant in the second class. The ASP program will select only one `match/2` predicate, i.e. either `match(c4,c5)` or `match(c4,c6)`.

Finally, line 12 aims to maximize associations specified by the `match/2` predicate concerning the first class (left term), corresponding to the number of pseudo-perturbations. The second class will have the same number of pseudo-perturbations due to the pairwise nature of matches. If we take the example in Figure 4.7A, two optimal equivalent solutions are possible: $\{(c1, c3)\}$ and $\{(c2, c3)\}$. However, the program retains

only one of them, but we have the possibility to recover all solutions through post-processing.

The complexity of our program can be analyzed considering two factors impacting the size of the search space: *(i)* the selection of k genes from a total set of G genes, and *(ii)* the choice of pairs of cells. That is, for each possible selection of k genes, an amount of c associations between cells in classes A and B (where the values of the k genes coincide) has to be tested to discard redundancies within the same class. The maximum value for c is $|A| \times |B|$; which represents associating all cells in both classes. The solver performs backjump and conflict-driven learning, optimizing the search space; thus, our estimate measures a worse case. The estimated complexity (see Equation 4.4) implies that our algorithm is exponential on the number of considered genes and cells from our scRNAseq dataset.

$$\mathcal{O}(\binom{|G|}{k} \times 2^{|A|\times|B|}) \tag{4.4}$$

where $|G|$ is the total number of genes, k is the number of genes to select and $|A|$ (resp. $|B|$) is the number of cells in class A (resp. class B).

4.3.2 Program application on different benchmarks

Our pseudo-perturbation identification program proposes an additional constraint, specifically aiming to ensure the generation of distinct pseudo-perturbations comprising k expressed genes within the same class. While increasing computational time, it proves valuable in handling redundant scRNAseq data.

Both programs were applied to datasets $A - P$ (see Table 4.2). For comparison purposes, we post-processed the results from Chebouba *et al.* [11] (C in Table 4.2) by removing redundant solutions (values in parentheses in Table 4.2). For further insights into dataset specific features, please refer to Table 4.1.

Optimal solutions were attained by both programs for datasets A and B (see Table 4.2). Suboptimal results are denoted with an asterisk (*) over the fixed timeouts. Our program (O in Table 4.2) yielded suboptimal results for datasets $C - P$, while Chebouba's program (C in Table 4.2) exhibited suboptimal outcomes for datasets D and P. Chebouba's version demonstrated shorter execution times than our version when no timeout was imposed. Analysis of the number of distinct pseudo-perturbations generated by each program reveals two different behaviors contingent on dataset nature and

complexity. Firstly, for single-cell datasets ($A-SC$), Chebouba's program computes either an equal or a smaller count of pseudo-perturbations when removing redundancies. For instance, for dataset C, Chebouba's program derives an optimal solution comprising only 1 distinct pseudo-perturbation, while our program yields suboptimal results, generating 6 and 11 distinct pseudo-perturbations for k = 3 or k = 10 respectively. This disparity is more pronounced in larger datasets like D (10 *vs.* 22) or SC (3 *vs.* 20). It indicates that Chebouba's program infers numerous redundant solutions, lacking the ability to differentiate them effectively, highlighting our program's superiority in handling scRNAseq data. Secondly, our version showcases superior outcomes for phosphoproteomics data (23 *vs.* 25), affirming its adaptability across single-cell or averaged cell population gene-expression datasets.

Table 4.2 – Comparison of ASP programs on different datasets.

Dataset	k	Execution time		Distinct Pseudo-Perturbations	
		C	O	C	O
A	3	0.008s	0.008s	3 (4)	3
B	3	0.048s	0.223s	1 (132)	4
C	3	1.420s	10 min*	1 (625)	6
	10	1.424s	10 min*	1 (600)	11
D	10	10 min*	10 min*	10 (2,436)	22
SC	10	5h 2 min	65h*	3 (77,618)	20
P	10	50h*	50h*	23 (64)	25

C corresponds to the Chebouba's logic program, while O corresponds to our logic program. For Chebouba's program, in parenthesis, the total number of pseudo-perturbations vectors (redundancy comprising). * Execution time corresponds to the fixed timeout.

4.3.3 Discrimination of the medium and late trophectoderm stages

PKN reconstruction We used 438 transcription factor (TF) genes involved in human embryonic development as input for pyBRAvo software [92] to reconstruct a PKN. These TF genes were identified through WGCNA [71] analysis of scRNAseq data (see Section 3.2.2 ; TF list in Appendix D). The exploration depth parameter was fixed to 2, i.e., up to 2 levels upstream of the initial TFs. Only gene transcription events were queried, yielding a PKN of 327 nodes and 475 edges, with only 28 of the 438 initial TFs found in the database. We then reduced the network to 191 nodes (84 input genes, 27 intermediate

genes, 14 readout genes, and 66 complexes) and 285 edges (Figure 4.8), limited to genes measured in scRNAseq data and complexes linked to these genes.

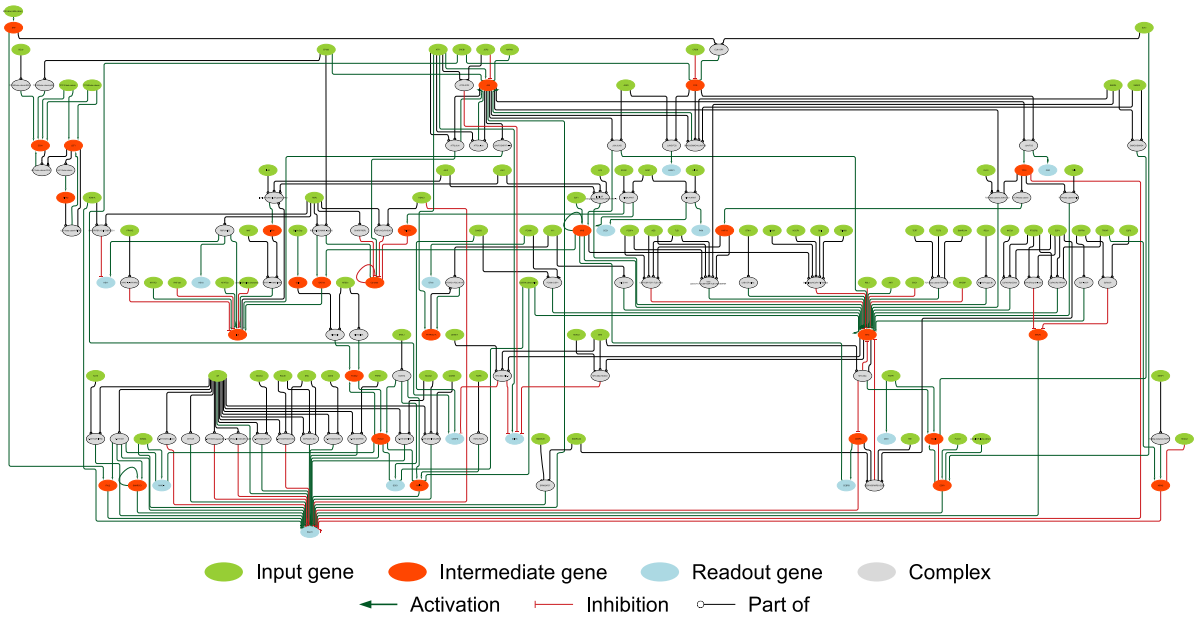


Figure 4.8 – Reconstructed Prior Knowledge Network.
A better quality version of the figure can be found in the Appendix E.

Experimental design construction We generated pseudo-perturbations for the experimental design using the method described in Section 4.2.5.1, which employed the set of input and intermediate genes from the reduced PKN, comprising 111 (84 + 27) genes. Our analysis focused on the expression of these genes across 680 cells, which were identified to be in medium and late TE developmental stages (see Table 4.1, dataset *SC*).

We tested different values of k, the number of selected genes, similar to those used in Videla *et al.* [99], the study introducing Caspo. We observed the number of pseudo-perturbations generated after 30 hours of calculation on a computer cluster and computed the representativity score (Equation 4.3) for each k value. Based on our results, $k = 10$ was the best trade-off between a high number of pseudo-perturbations and a high representativity score (see Figure 4.9). This value was also used in Chebouba *et al.* [11], supporting our decision.

Our method produced 20 pseudo-perturbation Boolean vectors, which paired medium and late TE cells to maximize the expression value difference of 14 readout genes. In Figure 4.10, we present the experimental design composed of 24 genes: 7 inputs genes

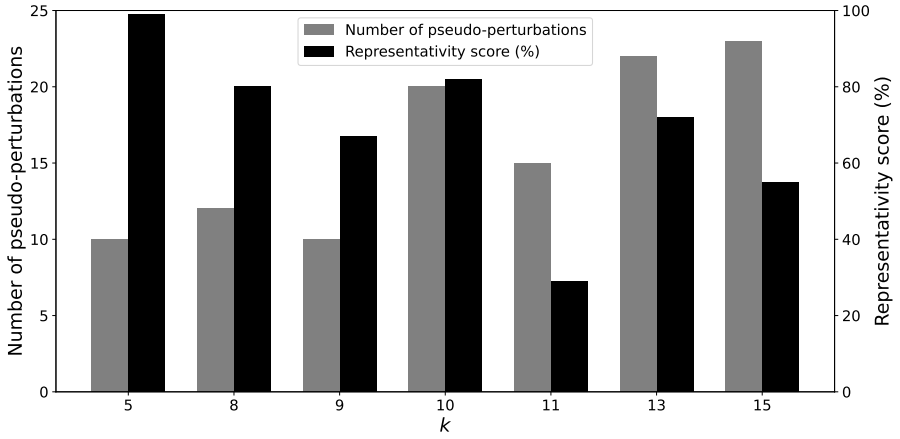


Figure 4.9 – Impact of k on the number of pseudo-perturbations and their representativity.

The number of pseudo-perturbations was found after 30 hours of calculation on a computer cluster. The representativity score is the mean of the two representativity scores calculated for both classes medium and late TE (see Equation 4.3).

(in green), 3 intermediate genes (in red), and 14 readouts (in blue). Each row represents a pseudo-perturbation (on the left, ordered from most to least representative) and its readout observations. Note that each vector is unique. We observe some readout genes with minimal variations (mean of expression difference between both stages less than 0.06), e.g., *DEC1* or *SOD1*, and some readout genes where a significant variation (mean of expression difference between both stages greater than 0.30) is observed, e.g., *CEBPB*, *CEBPD* or *GSR*. These last also appear in the learned BNs (see Figure 4.12). In Figure 4.11, a more detailed examination of the expression of readout genes in the learned BNs is presented. For instance, *CEBPB*, identified in the late TE BN, exhibits lower expression in medium TE compared to late TE. Conversely, *PSAT1*, present in both medium and late TE stages, shows minimal expression variations between the two stages.

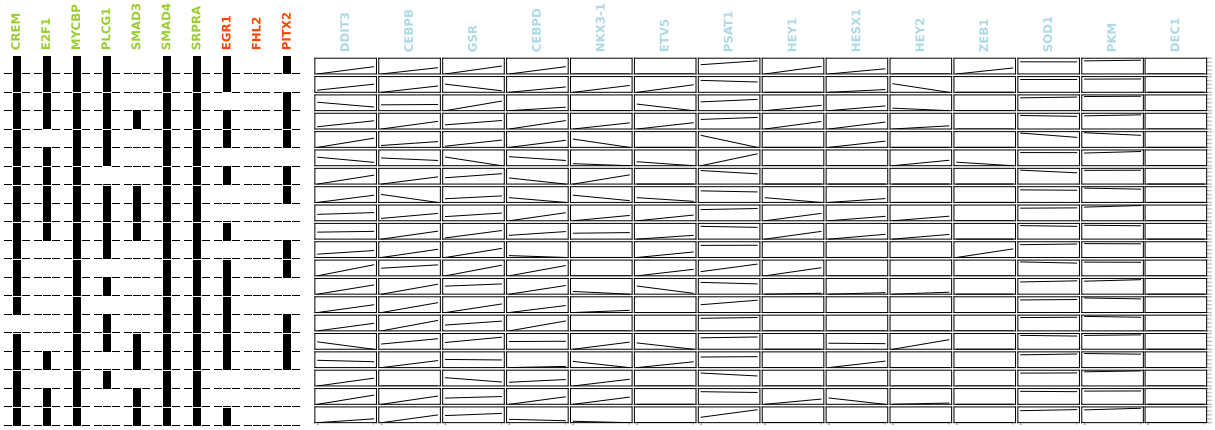


Figure 4.10 – Graphical representation of computed experimental designs. Each row (left side) represents an optimal pseudo-perturbation on the 10 selected input (green) and intermediate (red) genes. Binarized vectors are illustrated using bars, where a black (resp. white) bar means the gene is active (resp. inactive). On the right side, readout genes in blue, present in the inferred BNs (cf. Figure 4.12), are shown. In each box, the curve represents the normalized readout gene expression evolution between the medium TE (M, left) and late TE (L, right) developmental stages.

BN inference Using Caspo software [99], we infer families of BNs for medium and late TE using the reconstructed PKN and specific experimental designs. Caspo generates BNs adhering to PKN topology, optimizing the Mean Square Error (MSE) between Boolean predictions of readout nodes and experimental measurements. Our Caspo parameters are set as: (i) $length$ = 2 to restrict nodes receiving at most 2 "AND" logical functions, (ii) $fitness\_tolerance$ = 0.0001 to permit exploration beyond the optimal BN up to a distance of 0.01% from the optimal MSE, and (iii) size_tolerance = 0 to avoid any tolerance in size. The first parameter is fixed to simplify the exploration search. The second is set to strike a balance between strict exploration (searching for optimality only) and avoiding too much laxity. The third parameter is fixed to ensure the BNs are as minimal as possible in terms of size.

Figure 4.12 displays the learned BNs union for the two studied developmental stages. The size is equal to 8 for medium TE and 15 for late TE, with respective optimal MSEs of 0.1421 and 0.1924. The higher MSE for late TE indicates a more intricate fitting between inferred BNs and experimental data, resulting in less precise BNs compared to medium TE. While the medium TE family contains 2 BNs, the late TE family comprises 4. Notably, these BN families diverge in gene regulatory mechanisms. Late TE BNs display more extensive connectivity with 3 inputs and 4 readouts, compared to 2 inputs and 1 readout in medium TE. Both share 4 genes, including 2 inputs (*SMAD3* and *E2F1*), 1

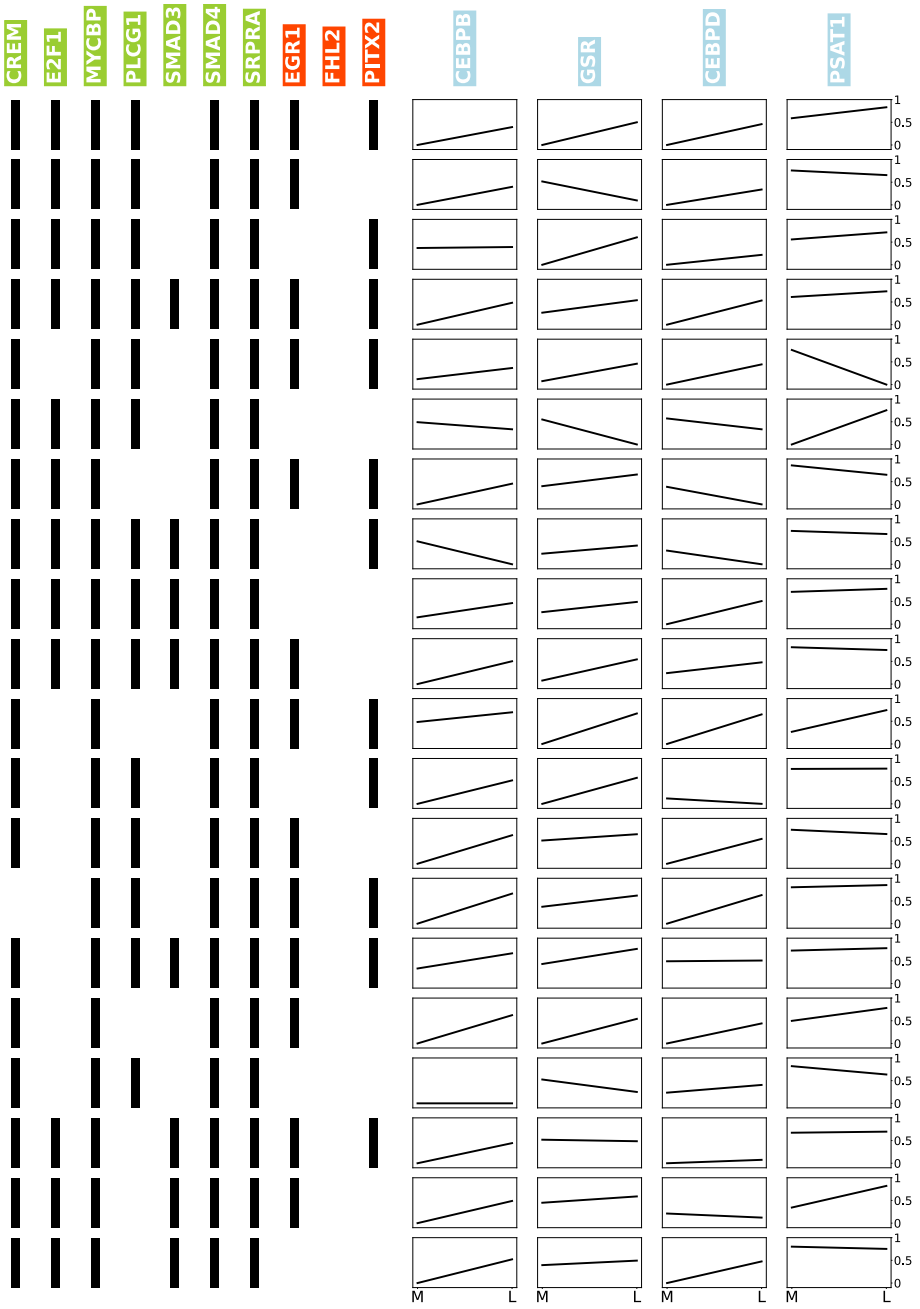


Figure 4.11 – Graphical representation of computed experimental designs focusing on readouts involved in inferred BNs.

Each row (left side) represents an optimal pseudo-perturbation on the 10 selected input (green) and intermediate (red) genes. Binarized vectors are illustrated using bars, where a black (resp. white) bar means the gene is active (resp. inactive). On the right side, readout genes in blue, present in the inferred BNs (cf. Figure 4.12), are shown. In each box, the curve represents the normalized readout gene expression evolution between the medium TE (M, left) and late TE (L, right) developmental stages.

intermediate (*EGR1*), and 1 readout (*PSAT1*). Late BNs exhibit supplementary readout genes, namely *GSR*, *CEBPB*, and *CEBPD*, indicating that the readout measurements matched the late TE BNs prediction, given the selected pseudo-perturbation Boolean vectors. However, medium TE BNs could not predict the observed measurements with minimal error on these three genes. This suggests higher complexity in late TE regulatory mechanisms compared to medium TE.

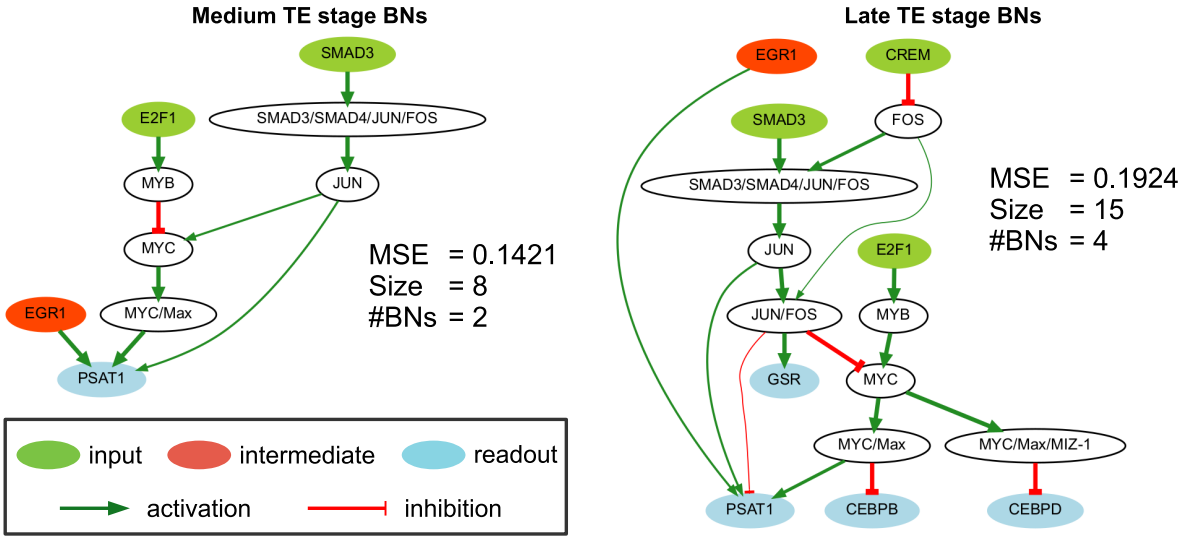


Figure 4.12 – Families of inferred Boolean networks (BNs) for medium and late TE developmental stages.
Each network represents the union of (sub-)optimal BNs learned from the reduced PKN and the experimental design. Recall that we set the fitness parameter to $fitness\_tolerance = 0.0001$, allowing the inference of suboptimal BNs with a superior MSE up to 0.1% of the optimal MSE. The colored nodes represent genes associated with experimental designs, including input and intermediates involved in pseudo-perturbations, and readout genes. The width of the arcs represents the frequency of occurrence of this arc in the BNs.

4.4 Discussion and conclusion

Studying human embryonic development poses challenges, often requiring system perturbations, which are infeasible here. To overcome this, we propose an original framework utilizing human embryonic scRNAseq data to identify pseudo-perturbations and construct Boolean network families representing two specific developmental stages.

As significant results, we developed an algorithm to obtain pseudo-perturbations

from scRNAseq data demonstrating scalability and efficiency through benchmarking with datasets of varying sizes. The worst-case search complexity for the real case study was of $\binom{111}{10} \times 2^{348\times332} = 3.26 \times 10^{34793}$ (see Equation 4.4), and our partial results were generated in 65h. We prove that our algorithm allows for more diverse pseudo-perturbation sets than the state-of-the-art method Chebouba *et al.* [11], which studied cell population-averaged measurements. We can simulate real perturbations by identifying pseudo-perturbations and proposing more precise (such as Boolean) computational models. Our method identified 20 pairs of cells with Boolean expressions coinciding with selected genes, representing of 75% and 89% of the complete set of cells in medium and late TE developmental stages, respectively.

To contextualize the retrieved set of 20 pseudo-perturbations, it is important to note that these 20 pseudo-perturbations represent a suboptimal result obtained after 65 hours of computation. If we keep running the program more pseudo-perturbations would be found. For this suboptimal solution, we exploit the cell redundancies via the readout maximization. This solution comprises $k = 10$ selected genes; however, this solution is not unique, as other subsets of k genes exist to identify 20 pseudo-perturbations. We compute a suboptimal count of equivalent solutions yielding more than 1 million of solutions. In the next chapter, we explore the space of equivalent solutions, in terms of similar selections of k genes and convergence in the number of pseudo-perturbations.

Using diverse pseudo-perturbations sets, we generate families of Boolean networks to distinguish medium and late TE developmental stages in human embryonic development. The BNs propose Boolean functions derived from the Pathway Commons database to model gene regulation mechanisms. Late TE cells exhibit a more complex BN structure (size 15 vs. 8) than medium TE cells. These findings are consistent with the fact that late TE requires a gain of biological function to help the embryo implant in the endometrium. Differently, from methods that propose a single computational model of averaged cells, our method includes a subset of 20 cells for each stage and learns optimal families of BNs representing the diversity of expression mechanisms within this cell subset for each stage.

Compared to other methods proposing computational models from scRNAseq data [62, 68], our method outputs logic models when no perturbation data is available, and uses a pool of cell behaviors to represent a single developmental stage. Extending this approach to other developmental stages could deepen our understanding of regulatory mechanisms in the human embryonic development. In addition, its adaptability makes it versatile for diverse case studies.

As perspectives from these results, we intend to study the impact of parameters used in the PKN reconstruction step, such as exploration depth or excluded databases. Meaningful interactions may be lost at this step. In addition, we only retrieve around 8% of the initial TFs for the reconstruction (28 out of 438), indicating a need for further research to enhance the PKN. Furthermore, the considerable storage and execution time requirements (65 hours for 20 pseudo-perturbations, with a memory footprint of 292 GB) highlight a need of improvement of the algorithm is required in order to expedite the process. Finally, we want to further investigate the exploration of equivalent solutions by potentially incorporating constraints inspired by biological hypotheses. These forward-looking perspectives form the crux of the subsequent chapter, where we refine our methodology and unveil novel insights.

Chapter 5

Contribution 2: More robust inferred Boolean networks using SCIBORG

"The task is, not so much to see what no one has seen yet; but to think what nobody has thought yet, about what everybody sees."

— Arthur Schopenhauer (1788-1860)

Summary

This chapter introduces SCIBORG, an improvement of the presented method in previous chapter allowing the learning of Boolean networks from singe-cell transcriptomic data and prior knowledge. The method and findings presented here will be the subject of an upcoming article to be submitted to a journal in the field of computational biology. SCIBORG builds upon the method discussed in Chapter 4. We present enhancements made to the ASP program for identifying pseudo-perturbations, enabling better results in less computational time. In this work, we investigate findings in detail, exploring equivalent solutions to provide more exhaustive results. We also introduce a cell classifier that sorts a given cell into a specific developmental stage. Focusing on medium and late trophectoderm (TE), we learn two distinct BN families and identified stage-specific regulatory mechanisms.

5.1 Introduction

Understanding the regulatory mechanisms involved in cell differentiation process is central to studying the human embryonic development. Such chain of events regulates human preimplantation development, leading to an implantation-competent embryo. Advanced technologies like transcriptomics enable deciphering regulatory events implicated in this system. This helps biologists to refine robust embryo quality assessment techniques, and potentially improve assisted reproductive technologies, such as in vitro fertilization, which has a low success rate of around 25%. Future advances in this field establishing computational models for preimplantation development that are truly useful for simulating the system.

Typically, system perturbations and their impacts are used to model a system. Dunn *et al.* [68] inferred mouse embryonic models from single-cell transcriptomic and knockout data on mouse stem cells, revealing insightful results via a perturbation-based approach. Another study by Chevalier *et al.* [62] learned ensembles of dynamical logic models of the cell fate differentiation implicated in tumor invasion and migration. This method did

not use system perturbations but utilized averaged cell expressions of each differentiation stage, providing an overview of the involved differentiation processes. In Moignard *et al.* [69], the authors use single-cell expression of tens of genes well characterized in the early blood development to infer Boolean networks. These BNs allowed the researchers to identify regulatory mechanisms of 20 transcription factors modeling the cell differentiation in the studied system.

In our study, perturbations are unfeasible due to various factors, such as ethical, biological or legal constraints. Moreover, we aim to include as much gene expression data as possible from the scRNAseq data we have. Our goal is to understand the underlying regulatory mechanisms leading from a stage to another in cell fate decisions. To investigate this question, we developed in Bolteau *et al.* [5] a method to infer Boolean networks modeling two developmental stages, distinguishing them by discrepancies in involved regulatory mechanisms. One primary concern of this method is the computation time and memory required to obtain results. To overcome this, we introduce SCIBORG.

SCIBORG is an improvement of our previously presented findings, significantly reducing execution time and memory usage. It explores outcomes in detail, enabling the production of more exhaustive and robust results and offers the possibility to identify regulatory mechanisms involved in specific developmental stages. Furthermore, our method could be extended to be applied on different developmental stages and various biological studies.

In this study, we undertook the Boolean network (BN) inference as a Boolean satisfiability problem relying on ASP. Focusing on medium and late trophectoderm (TE) developmental stages, we identified 96 pseudo-perturbations compared to 20 in previous studies, in less computational time, from single-cell transcriptomic data. Leveraging prior knowledge from databases and these identified pseudo-perturbations, we learned stage-specific BN families. The inferred BNs are robust and suggest exhaustive identification of stage-specific regulator mechanisms.

5.2 Improvements of the method

In this section, we present enhancements and key modifications of our method presented in Chapter 4. Thus, the improved method, providing more enxhausive and robust results, is named SCIBORG for *using Single-Cell data to Infer BOolean networks modeling Regulation of Genes*. In a nutshell, SCIBORG comprises the same steps: *(i)* PKN

reconstruction, *(ii)* experimental design construction, and *(iii)* BN inference. Regarding data preprocessing, we explore a new normalization method using an arctangeant function. Additionally, we extend our investigation by employing a larger PKN, achieved by increasing the reconstruction depth for pyBRAvo. As demonstrated previously, the efficiency of our ASP program for pseudo-perturbations identification leads to an improvement in terms of storage and computational time. Here, we refine this program by adjusting some rules and constraints. Furthermore, we delve into other aspects of our results by analyzing equivalent solutions for k selected genes and a specific number of pseudo-perturbations. Additionally, we introduce a *cell classifier* that enables cell classification.

5.2.1 Preprocessing

We consider 3 types of genes: inputs, intermediates and readouts, based on the PKN topology. Inputs and intermediates are binarized following the formula presented in previous chapter (see Equation 4.1). Readout genes undergo normalization. We explore two normalization techniques to scale the expression values to the range $[0, 1]$:

1. a "min-max" normalization:
$$n_{ij} = \frac{r_{ij} - r_{min}}{r_{max} - r_{min}} \tag{5.1}$$
2. an "arctangeant" normalization:
$$n_{ij} = \frac{2}{\pi} \times \arctan(r_{ij}) \tag{5.2}$$

Here, in both equations, n_{ij} represents the normalized expression of the gene j for the cell i and r_{ij} denotes the raw expression of the gene j for the cell i. For "min-max" normalization, r_{min} (resp. r_{max}) is the minimum (resp. maximum) expression value of all readout genes across all cells involved in the studied developmental stages.

In Figure 5.1A, we present the functions of the two normalizations, with arbitrary values $r_{min} = 0$ and $r_{max} = 200$ for "min-max" normalization. Given these functions, the number of intermediate values (around 0.5) is more important for "min-max" normalization, which is a linear function. This phenomenon becomes apparent when analyzing the distribution of normalized expression of readout genes involved in the PKN^0 (see Section 5.3.2). Numerous genes have intermediate expression considering the "min-max" normalization, while for "arctangeant" normalization, expression values are closer

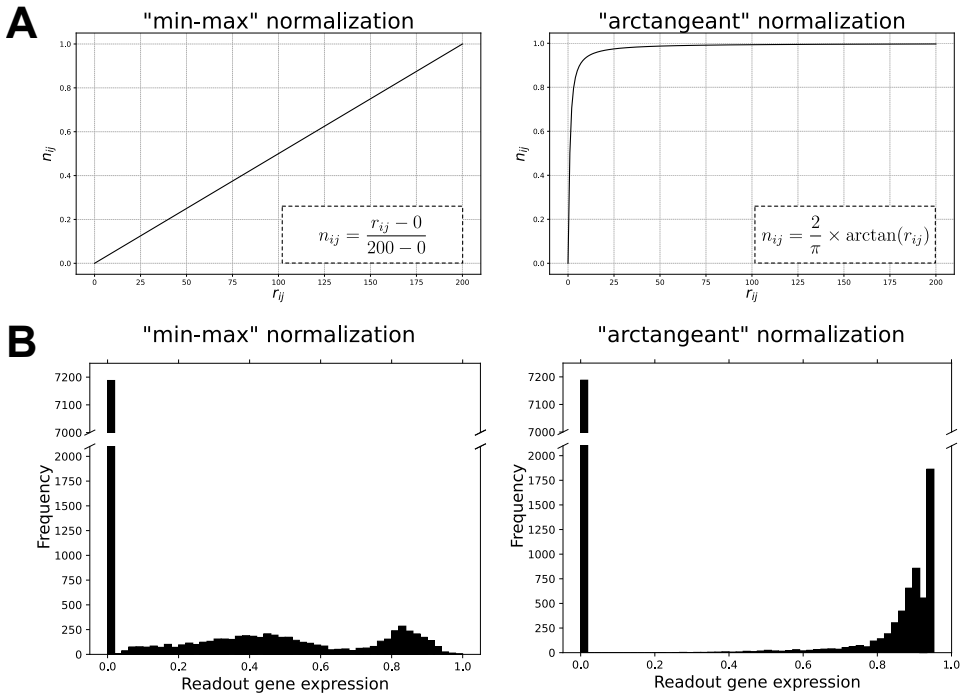


Figure 5.1 – Comparison of "min-max" and "arctangeant" normalizations.
(A) Normalization functions represented graphically. On the left, the"min-max" normalization following Equation 5.1 with r_{min} = 0 and r_{max} = 200, and on the right, "arctangeant" normalization following Equation 5.2.
(B) Readout gene expression frequency of the two explored normalizations. On the left, the"min-max" normalization distribution and on the right, "arctangeant" normalization distribution. We consider for these distributions the 19 readout genes involved in the PKN^0 (see Section 5.3.2), expressed in the 680 cells across medium and late TE developmental stages.

to 0 or 1 (Figure 5.1B).

Recall that Caspo predicts readout values through inferred BNs and predicts either 0 or 1 value for each readout (see Section 3.4.2). Given a prediction of Caspo (0 or 1), the distribution of "min-max" normalization implies, in average, a small distance between the prediction and the real value. In contrast, for "arctangeant" normalization, the distance will be larger if the prediction does not fit well with the real value, e.g., prediction of 0 and real value around 1.

Furthermore, "min-max" normalization is specific to readouts and cells: the minimum and maximum can vary if we consider other developmental stages or readouts. This could lead to data-dependant outcomes, which is not the case with "arctangeant" normalization.

5.2.2 PKN reconstruction

As observed previously, the PKN reconstruction relies on pyBRAvo software. Thanks to our improvements in pseudo-perturbations identification program (see Section 5.2.3), SCIBORG enables the use of larger PKNs. To achieve this, we leverage the reconstruction depth parameter (see Section 3.4.1). Recall that the reconstruction depth parameter allows the algorithm to recursively search for predecessors until the fixed depth. We test two values of the depth parameter: 0 and 2. Here, a depth equal to 0 signifies "infinite" depth, resulting in the reconstruction of the largest possible PKN from the queried Pathway Commons database given a initial gene list. Throughout the manuscript, the PKNs reconstructed with these parameters are denoted PKN^0 and PKN^2, respectively. Afterwards, the PKN is reduced similarly to the previous contribution (see Section 4.2.4.2).

In Section 5.3.2, we present the outcomes generated by the parameter value and PKN reduction modifications. We also compare this new PKN^0 with the one previously reconstructed (PKN^2).

5.2.3 Pseudo-perturbation identification improvements

We refine the ASP program for the identification of pseudo-perturbations, allowing us to retrieve more pseudo-perturbations in less time than the previous version. In the remainder of this manuscript, we will refer to the first version of the ASP program, presented in previous chapter in Section 4.3.1, as version $v1$. The new version of the ASP program presented in this chapter will be referred to as version $v2$. In Section 4.2.5.1, we introduce 3 constraints to identify pseudo-perturbations for $v1$. In this version $v2$ of the program, we refine the first constraint and create a fourth one.

Problem statement We use a binary matrix E, where e_{ij} represents the presence or absence of gene j for cell i. We denote by C the set of cells and by G the set of genes. Each cell in C is associated with one class, either A or B; thus $C = A \uplus B$. We define the relation I^G, $I^G(c_i) = \{g_j \in G | e_{ij} = 1\}$, where $I^G(c_i)$ represents the active genes, belonging to G, for cell c_i. If $G' \subset G$, then the restriction of I^G to G' is defined by $I^{G'}(c_i) = I^G(c_i) \cap G'$. Let us define the function type_of(g_i), which provides the type of the gene g_i that can be: 'input', 'intermediate', or 'readout'. We define also the function parent(g_i, g_j), meaning that a path exists from the g_i to the gene g_j, regarding the network topology of the PKN.

Problem formulation Given a matrix E, a parameter k limiting the number of selected genes, and a parameter l giving the proportion of input genes in the k selected genes, find a subset G' of genes and the largest subset C' ($C' = A' \uplus B' \subset C$, where $A' \subset A$ and $B' \subset B$) satisfying the following four constraints:

Constraint 1. The size of G' is fixed to k, where $|\{g_i \in G', \text{type\_of}(g_i) = \text{'input'}\}| = l$, with $g_i \in G'$.

Constraint 2. $\forall c_1, c_2 \in A'$ (resp. B'), $c_1 \neq c_2$, such that $I^{G'}(c_1) \neq I^{G'}(c_2)$.

Constraint 3. $\forall c_1 \in A'$ (resp. B'), $\exists c_2 \in B'$ (resp. A'), such that $I^{G'}(c_1) = I^{G'}(c_2)$.

Constraint 4. $\forall g_1$, with $\text{type\_of}(g_1) = \text{'intermediate'}$, $\exists g_2$, with $\text{type\_of}(g_2) = \text{'input'}$, such that $\text{parent}(g_2, g_1)$.

Constraint justification

Constraint 1. This constraint reduces the search space to improve the computational efficiency. We add a second parameter corresponding to the proportion of input genes in the k selected genes. In others words, when we select k genes, we ensure that a proportion l, where $l < k$, are input genes. The remaining genes $(k - l)$ will be intermediates. This ensures that both input and intermediate genes are included in the k selected genes.

Constraint 2. The second constraint prevents redundancy within a same class by forbidding cells from having the same gene expression for the k selected genes.

Constraint 3. The third constraint ensures a match between two identified cells from different class, having the same gene expression for the k selected genes.

Constraint 4. The last constraint ensures connectivity between inputs and intermediates. We add this constraint in order to have set of k genes that are connected to each other and potentially yielding the inference of input-intermediate-readout connected BNs.

5.2.4 Exploring equivalent solutions

The ASP program of pseudo-perturbation identification aims to maximize the number of pseudo-perturbations, resulting in a solution comprising a specific number of pseudo-perturbations for a chosen subset of k selected genes. It is important to note that this solution is not unique. Indeed, other subsets of k genes could yield the same number

of pseudo-perturbations. In such cases, the cells involved in pseudo-perturbations might differ from those in the initial solution. However, all these solutions are equivalent and could provide valuable insights. In Section 5.3.3.1, we consider equivalent solutions and use them to infer BNs.

5.2.5 Cell classifier

Recall that the BN inference using Caspo is based on predicting readout expression through the BN. The metric used for this prediction is the mean square error (MSE), which represent the distance between readout prediction values and observed expression values (see Section 3.4.2). We use this MSE metric to implement a *cell classifier*, aiming to sort cells into a class or, in our case, a developmental stage.

Our classifier follows this procedure:

1. **Distance computation:** Considering two classes, A and B, for each class, the algorithm computes the average MSE of readout predictions for the studied cell. This yield two distances representing the discrepancy between readout predictions and observations.

2. **Classification:** The algorithm assigns to the studied cell the class, either A or B, with the smallest distance.

Given a set of cells we want to sort, we use the cell classifier and calculate the following metrics:

- *Class-specific accuracy:* provides the percentage of accuracy (i.e., correct classification) for a specific class.

- *Global accuracy:* provides the average accuracy between the two studied classes.

- *Balanced accuracy (BAC):* provides a balanced accuracy considering the number of cells within each class.

5.3 Results

5.3.1 ASP program

5.3.1.1 Detailed description of the pseudo-perturbation identification program

In this section, we provide a step-by-step description of the version $v2$ program of pseudo-perturbation identification. We compare each part of the program with the corresponding part in the version $v1$.

Some rules and constraints are modified between the two versions. We also replaced some choice rules with constraints that are less costly in terms of memory and computation time. Recall that the program objective is to maximize the count of pseudo-perturbation matches between different class cells, respecting rules and constraints outlined in Section 5.2.3. Here, a match refers to identical Boolean expression values for a set of genes between two cells from different classes. We refer to these Boolean values as a pseudo-perturbation.

For a better visualization, we use two distinct background color for the following program codes, a purple one for $v2$ and a blue one for $v1$.

First, experimental data is formulated using `pert/4` predicates, giving the expression of a gene G at value S in cell C, linked to class CL. The version $v2$ program starts by selecting a set of l input genes from all genes that are not intermediate ones using the line $v2\_1$. Line $v2\_2$ selects a set of $k - l$ intermediates genes. Together, these rules ensure respect for the *Constraint 1* expressed in Section 5.2.3.

```
v2_1  {selinput(G) : pert(C,G,S,CL), not intermediate(G)} = l.
v2_2  {selinter(G) : intermediate(G)} = k-l.
```

In contrast, the version $v1$ selects k genes without any proportion of input or intermediate (line $v1\_1$), which can conduct to the selection of a set composed only of inputs or intermediates.

```
v1_1  {selgene(G) : pert(C,G,S,CL)} = k.
```

Once genes are selected, we filter experimental data using the `selpert/4` predicate. For each selected input (resp. intermediate), a predicate is defined in line $v2\_3$ (resp. $v2\_4$).

```
v2_3  selpert(E,V,S,C) :- selinput(V), pert(E,V,S,C).
v2_4  selpert(E,V,S,C) :- selinter(V), pert(E,V,S,C).
```

Similarly, version $v1$ summarizes experimental data focusing on global selected genes (line $v1\_2$).

```
v1_2  selpert(C,G,S,CL) :- selgene(G), pert(C,G,S,CL).
```

Then, line $v2\_5$ associates with the `equal/3` predicate a pair of cells I and J having the same expression for the gene G (`S1==S2`). Line $v2\_6$ defines a potential match (`pot_match/2` predicate) between two distinct cells I and J (`I!=J`) if there exist k genes having the same expression for these two cells (`equal/3` predicate). These cells should be part of a different class (`C1<C2`). This rule means that cells involved in a `pot_match/2` predicate could form a match because they have the same expression for the k selected genes. A potential match could lead to a match or not. This is the purpose of the line $v2\_7$ where, given a `pot_match/2` predicate, a `match/2` can be inferred or not, via the choice rule `0{...}1`.

```
v2_5  equal(I,J,G) :- selpert(I,G,S1,C1), selpert(J,G,S2,C2), I!=J,
        S1=S2.
v2_6  pot_match(I,J) :- k = {equal(I,J,_)}, selpert(I,_,_,C1),
        selpert(J,_,_,C2), C1<C2, I!=J.
v2_7  0{match(I,J)}1 :- pot_match(I,J).
```

The version $v1$ uses similar process. First, it calculates `equal/3` predicates referring to a pair of cells having similar expression for a given gene. Then, the program counts for a pair of cells the number of genes where expression is identical for both cells (line $v1\_4$). It stores the value by inferring a `countequal/3` predicate. Finally, a `match/2` predicate is inferred or not, for all pairs of cells where the similarity count is equal to k (line $v1\_5$).

```
v1_3  equal(I,J,G) :- selpert(I,G,S1,C1), selpert(J,G,S2,C2), C1<C2,
        S1=S2.
v1_4  countequal(I,J,M) :- M = {equal(I,J,_)}.
v1_5  0{match(I,J)}1 :- countequal(I,J,k).
```

Here, similar processes are used for both program versions. However, version $v2$ creates a predicate of arity 2 (`pot_match/2`), while version $v1$ employs a predicate of arity 3 (`countequal/3`). This bigger arity for $v1$ demands more resources in terms of execution

and storage, rendering version $v2$ less ressource-intensive.

Afterwards, the $v2$ program handles data sparsity by defining `nbInputOnes/2`, counting for a cell C involved in a match, the number of expressed input genes. With the constraint line $v2\_9$, the program forbids a match comprising a cell having less than 1 expressed input. Similar lines compose the $v1$ program (code not shown, please refer to Program 4.1). Together, both programs ensure that each pseudo-perturbation has at least one expressed input gene (referred to as *Constraint 3* in Section 5.2.3).

```
v2_8  nbInputOnes(C,N) :- N = {pert(C,G,1,_) : selinput(G), input(G)},
          match(C,_).
v2_9  :- match(C,_), nbInputOnes(C,N), N < 1.
```

Lines $v2\_10$ to $v2\_13$ answer to *Constraint 2* of Section 5.2.3, by preventing potential redundancies in a cell class. The objective is to forbid the selection of cells within the same class having the same k-genes expression. For that, we implement two constraints (lines $v2\_10$ and $v2\_11$) that forbid solutions where the same cell is present in two `match/2` predicates, in the first class (resp. second class) with line $v2\_10$ (resp. $v2\_11$).

```
v2_10  :- match(I,J1), match(I,J2), J1!=J2.
v2_11  :- match(I1,J), match(I2,J), I1!=I2.
v2_12  :- pot_match(I1,J), pot_match(I2,J), match(I2,J), I1<I2.
v2_13  :- pot_match(I,J1), pot_match(I,J2), match(I,J2), J1<J2.
```

To better illustrate this, let us consider a toy example comprising 3 cells in class A and 3 cells in class B following the "V-patterns" presented in Figure 5.2A. The cell $a1$ may match with $b1$ and $b2$, and cell $b3$ may match with $a2$ and $a3$, representing in total 4 potential matches. Given the line $v2\_7$, a potential match could infer a match for the considered pair of cells. Thus, $2^4 + 1$ possible solutions could be deduced. The additional solution corresponds to the empty set, which signifies inferring zero matches. An example of an induced solution could be the configuration where all 4 potential matches are inferred as matches (Figure 5.2A). However, this configuration contains redundancies that must be avoided. This is achieved with the constraints lines $v2\_10$ and $v2\_11$ forbidding V-patterns in all solutions. Therefore, the program generates 9 possible solutions having 0, 1 or 2 matches. We clarify this example pattern using an illustration notebook available via

the link in footnote [1]. In this notebook, we show the ASP results provided by the program on the toy example of Figure 5.2A.

The two constraints in lines $v2\_10$ and $v2\_11$ prohibit redundancies of V-patterns; however, one configuration does not comply with what is required. This configuration is created when multiple cells within a class match with multiple cells within the other class. We illustrate in Figure 5.2B the simpler case where 2 cells of a class have the same k-gene expression of 2 cells of another class. In this case, we have 4 potential matches, but we want to yield only one of them because they are all equivalent. After applying the $v2\_10$ and $v2\_11$ rules, we obtain 2 patterns comprising 2 matches: the "parallel-pattern" and the "cross-pattern". To overcome this, we introduce two constraints (lines $v2\_12$ and $v2\_13$). These two constraints are complementary. The first, in line $v2\_12$, prohibits a solution where 2 cells ($I1$ and $I2$) potentially match with a cell J and a match between $I2$ and J is considered. The idea behind this is to only keep the match between $I1$ and J, by filtering cross- and parallel-patterns of $I2$. The second, in line $v2\_12$, is similar to the first one, filtering cross- and parallel-patterns of the opposite symmetry. These constraints reinforce to keep on this type of configuration (B), only 1 match: `match(a1,b1)`.

We illustrate these rules and constraints considering the configuration B in an online notebook accessible via the link in footnote [2]. With this notebook, we can manipulate rules and constraints with the proposed toy example and better understand the concepts.

Two remarks are necessary. First, if the constraints $v2\_12$ and $v2\_13$ are applied to configuration A (V-patterns), only 1 solution will be admissible (out of the 9 possible after $v2\_10$ and $v2\_11$): `{match(a1,b1), match(a2,b3)}`. Second, it is important to note that our program is very constraining, drastically limiting the number of equivalent solutions. While other possible matches between cells exist, they are redundant. We explore these redundancies afterwards using a Python program in a later step called "readout difference maximization" (see Chapter 4, Section 4.2.5.2).

Program $v1$ handles redundancies by identifying differences of expression (`S1=!S2`) for the gene G between two cells $I1$ and $I2$ within the same class (`C1==C2`), referred as `diff/3` predicate (line $v1\_8$). It generates a predicate `countdiff/3` for each pair of cells $I1$ and $I2$ (line $v1\_9$). Finally, combined line $v1\_10$ and line $v1\_11$, the program forbids the

1. https://mybinder.org/v2/gh/mathieubolteau/Bolteau_PhD_Thesis_Supplement/master - see Notebook `Illustration_5.1.ipynb`. The web page may take a few minutes to load.
2. https://mybinder.org/v2/gh/mathieubolteau/Bolteau_PhD_Thesis_Supplement/master - see Notebook `Illustration_5.1.ipynb`. The web page may take a few minutes to load.

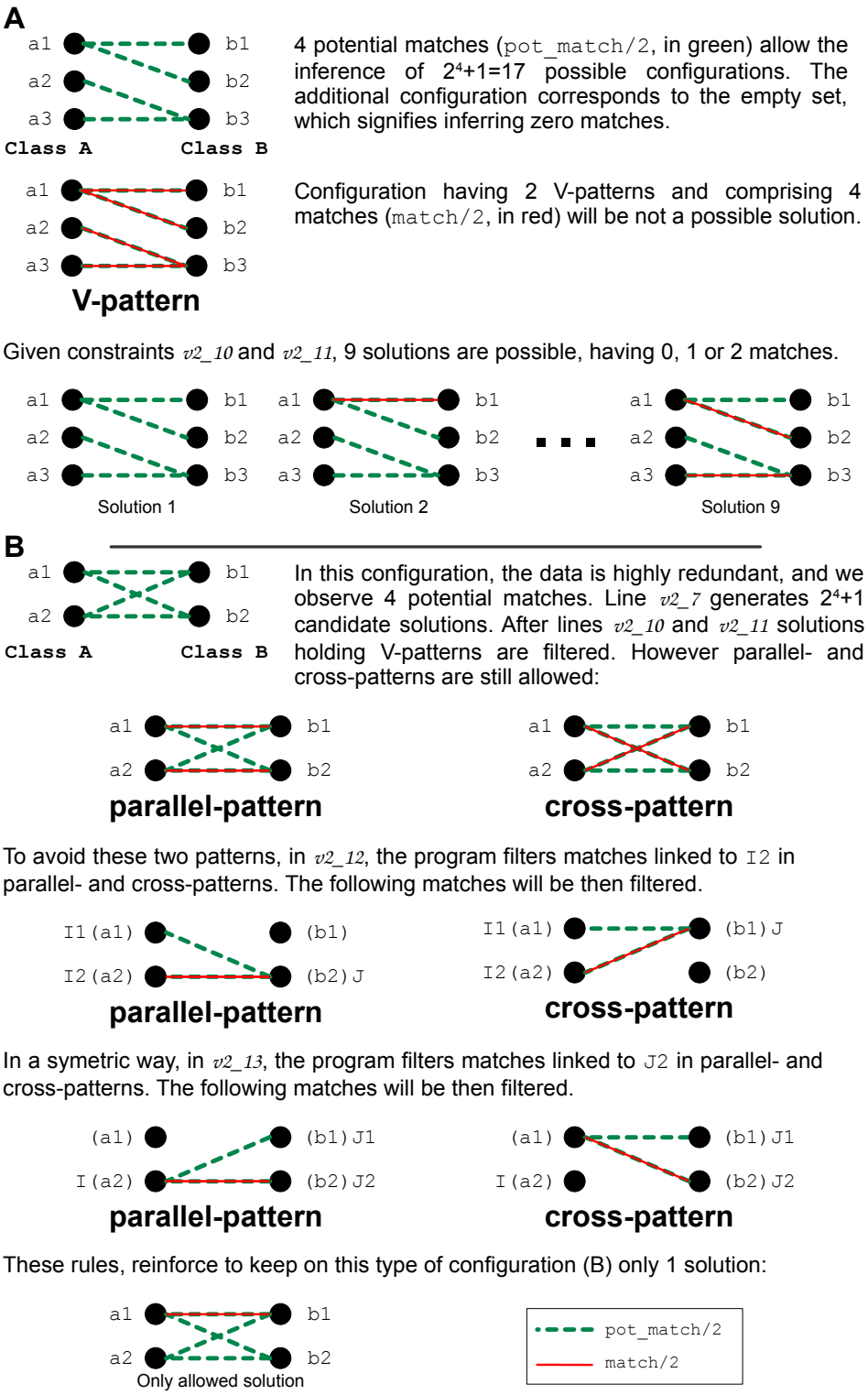


Figure 5.2 – Illustration of the *Constraint 2*.

(caption on next page.)

presence of `match/2` predicate where cell $I1$ and $I2$ have no gene expression differences (`countdiff(I1,I2,0)`).

```
v1_8  diff(I1,I2,G) :- selpert(I1,G,S1,C1), selpert(I2,G,S2,C2), C1=C2,
        S1!=S2, I1<I2.
v1_9  countdiff(I1,I2,M) :- M = {diff(I1,I2,_)}.
v1_10 :- countdiff(I1,I2,0), match(I1,_), match(I2,_), I1<I2.
v1_11 :- countdiff(I1,I2,0), match(_,I1), match(_,I2), I1<I2.
```

Comparing $v1$ and $v2$ for the redundancies handling, the first version introduces 2 new predicates that are costly for the solver, especially the `countdiff/3` one which requires a costly choice rule. Note that the grounder needs to enumerate all possible sets that respect the rules in a choice rule, which is highly resource-consuming in terms of execution time and storage. Version $v2$ uses only constraints that are not costly for the solver and allows by filtering to improve the solving process.

The line $v2\_14$ of the program $v2$ implements the *Constraint 4* presented in Section 5.2.3. The rule ensures to have, given a selected intermediate gene I, at least 1 selected input G topologically predecessor of I (`parent(G,I)`) using a choice rule `1{...}`.

```
v2_14 1{selinput(G) : parent(G,I), input(G)} :- selinter(I).
```

Finally, line $v2\_15$ aims to maximize the number of matches comprising cells involved in a pseudo-perturbation. The same computation is done in version $v1$, consequently we do not show it here (refer to Program 4.1).

```
v2_15 #maximize{1, I : match(I,_)}.
```

We provide in Program 5.1 the whole ASP program of the version $v2$. Without considering the new parameter l and the input-intermediate connectivity rule, version $v2$ contains 6 rules, including 4 choice rules, 5 constraints and 1 optimization, while version $v1$ comprises 8 rules, including 5 choice rules, 3 constraints and 1 optimization.

(continued from previous page.)
The *Constraint 2* prevents redundancies in pseudo-perturbations through the lines $v2\_10$ to $v2\_13$ in ASP program $v2$. Diverse patterns contain redundancies and need to be forbidden.
(A) Prohibition of V-patterns illustrated with a toy example.
(B) Prohibition of parallel- and cross-patterns illustrated with a toy example.

```
1  {selinput(G) : pert(C,G,S,CL), not intermediate(G)} = l.
2  {selinter(G) : intermediate(G)} = k-l.
3  selpert(E,V,S,C) :- selinput(V), pert(E,V,S,C).
4  selpert(E,V,S,C) :- selinter(V), pert(E,V,S,C).
5  equal(I,J,G) :- selpert(I,G,S1,C1), selpert(J,G,S2,C2), I!=J, S1=S2.
6  pot_match(I,J) :- k = {equal(I,J,_)}, selpert(I,_,_,C1),
       selpert(J,_,_,C2), C1<C2, I!=J.
7  0{match(I,J)}1 :- pot_match(I,J).
8  nbInputOnes(C,N) :- N = {pert(C,G,1,_) : selinput(G), input(G)},
       match(C,_).
9  :- match(C,_), nbInputOnes(C,N), N < 1.
10 :- match(I,J1), match(I,J2), J1!=J2.
11 :- match(I1,J), match(I2,J), I1!=I2.
12 :- pot_match(I1,J), pot_match(I2,J), match(I2,J), I1<I2.
13 :- pot_match(I,J1), pot_match(I,J2), match(I,J2), J1<J2.
14 1{selinput(G) : parent(G,I), input(G)} :- selinter(I).
15 #maximize{1, I : match(I,_)}.
```

Program 5.1 – Version $v2$ of the pseudo-perturbation identification program.

5.3.1.2 Identification of pseudo-perturbations across different datasets

We apply versions $v1$ and $v2$ on the five datasets A, B, C, D and SC presented in Section 4.2.3.1, Table 4.1. We show in Table 5.1 the obtained results regarding grounding and solving steps. For the grounding, we analyze the computation time and the memory size of this process. For the solving, we observe the solving time (for datasets C, D, we fix a timeout) and the number of pseudo-perturbations found by the program. Additionally, we compared the entire programs, noting that $v2$ includes additional rules to maintain *(i)* a proportion of inputs and intermediates (k and l parameters), and *(ii)* a connectivity between inputs and intermediates. Thus, the $v2$ program is more constraining, potentially leading to a selection of pseudo-perturbations different from the $v1$ program.

We can observe that the grounding memory of $v1$ is greater than $v2$. The significant difference for large datasets (D, SC), reaching a factor of more than 6 for D, is due to the

Table 5.1 – Comparison of the two program versions for identifying pseudo-perturbations.

Dataset	k	$l^{\dagger}$	Grounding Execution time		Grounding Memory size		Solving Execution time		Pseudo-perturbations count	
			$v1$	$v2$	$v1$	$v2$	v_1	$v2$	$v1$	$v2$
A	3	2	0.00s	0.01s	47 KB	26 KB	0.007s	0.009s	3	3
B	3	2	0.06s	0.06s	1.1 MB	451 KB	0.210s	0.060s	4	3
C	3	2	0.93s	0.26s	44 MB	6.2 MB	15min*	1.138s	7	6
	10	6	1.09s	0.25s	44 MB	6.2 MB	15min*	15min*	10	13
D	10	6	23.27s	5.75s	1.2 GB	186 MB	15min*	15min*	27	35
SC	10	6	793.46s	304.40s	18.41 GB	4.83 GB	65h*	7h*	20	92

† Considered parameter only for version $v2$. * Execution time corresponds to the fixed timeout.

large quantity of ground predicates generated during the grounding process, mainly caused by the choice rules of the $v1$ program. The grounding execution times are similar for small datasets (A, B), while for the other datasets (C, D, SC), $v2$ is at least twice as fast as $v1$. Despite the additional rules, $v2$ is less time-consuming than $v1$. Regarding solving, the execution times for datasets A and B are similar between the two versions. For the artificial dataset A, both versions identified the same number of pseudo-perturbations. For dataset B, where optimal results were found (no timeout), $v2$ identified one fewer pseudo-perturbation than $v1$ (6 vs. 7), due to the additional constraints in $v2$ on input and intermediate composition. A similar difference is observed for dataset C with k = 3, where $v2$ found the optimal number of pseudo-perturbations (6), while $v1$ identified a suboptimal count (7 pseudo-perturbations, with a 15-minutes timeout). For dataset C with k = 10 and dataset D, $v2$ outperformed $v1$ in the number of pseudo-perturbations found. The results for dataset SC are less comparable in terms of solving execution time; however, $v2$ found 4 times more pseudo-perturbations than $v1$ (20 vs. 92) with significantly less execution time.

To summarize, despite the additional constraints about the input-intermediate composition, $v2$ produces similar or better results than $v1$, with a shorter computation times, thereby significantly improving our overall method.

5.3.2 Reconstructed PKN

SCIBORG requires a PKN composed of signed interactions among genes representing prior-knowledge. We reconstruct our PKN using pyBRAvo from a list of 438 genes (see Appendix D), with the parameter *exploration_depth* = 0, indicating an "infinite" depth.

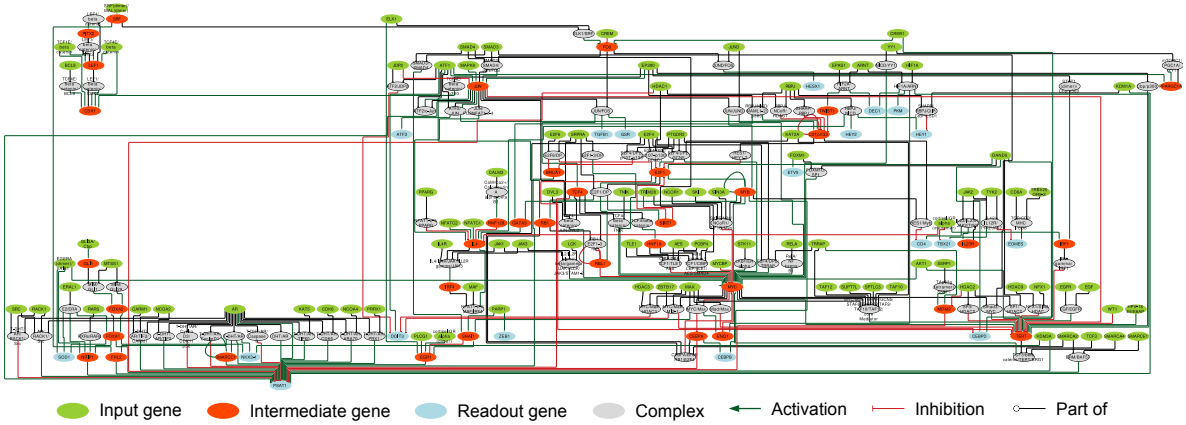


Figure 5.3 – Reconstructed Prior Knowledge Network PKN^0.
A better quality version of the figure can be found in the Appendix F.

This PKN comprising 435 nodes and 719 edges. Along these nodes, we identify 229 input genes, 66 intermediate genes, 21 readout genes and 119 protein-complexes. From the 438 genes given in input, 28 were retrieved within the PKN. After the reduction (see Section 5.2.2), the PKN, called PKN^0, comprises 225 nodes and 369 edges (Figure 5.3). More details on node composition are given in Table 5.2.

Table 5.2 – Comparison of the two studied PKN PKN^2 and PKN^0.

PKN	#nodes*	#edges	#inputs	#intermediates	#readouts	#protein-complexes
PKN^2	191 (28)	285	84	27	14	66
PKN^0	225 (28)	369	85	36	19	85

* The value in parentheses corresponds to the number of genes given to pyBRAvo found in the PKN.

We compare PKN^0 and PKN^2 in Table 5.2. As expected, PKN^0 is larger than PKN^2 in terms of number of nodes and edges. PKN^0 contains more prior knowledge allowing us to model better the biological facts that occur during embryonic development. The 121 input and intermediate genes within PKN^0 form the entry of the pseudo-perturbations identification program (see Section 5.3.3.1). Furthermore, we did not use PKN^0 with the initial method version, using the ASP program $v1$, due to a scaling up issue that prevented the use of a larger PKN.

5.3.3 Experimental design construction

5.3.3.1 Pseudo-perturbation identification

Our ASP program requires two parameters to identify pseudo-perturbations. The first is k, the number of input and intermediate genes we aim to select for establishing the pseudo-perturbation. The second is the value l fixing the number of input genes ($l \leq k$).

k parameter exploration Given the set of 121 input and intermediates genes in PKN^0, we executed $v2$ ASP program with different parameters. We explored different values of k ranging from 5 to 15. For each test, we fixed l to constitute 60% of the k selected genes, rounded down to the nearest whole number. The pseudo-perturbation identification program was run on a computer cluster during 30h stopping the solving after this timeout. The results are summarized in Figure 5.4A. We can observe that the number of pseudo-perturbations fluctuates with k, reaching a maximal value of 92 pseudo-perturbations for $k = 10$. This significantly increases the count of identified pseudo-perturbations compared to the results obtained with $v1$ ASP program (see Section 4.2.5.1). In addition, we calculated the cell representativity for each test, following Equation 4.3. Globally, cell representativity decreases as k increases (Figure 5.4B). With Caspo, more experimental data enable us to identify more regulatory mechanisms. Consequently, we select $k = 10$ and $l = 6$ parameters. Although the outcomes for $k = 12$ are also high in terms of identified pseudo-perturbation number (87), our goal is to maintain the highest possible representativity. Therefore, the results for $k = 10$ are more suitable for our purposes when comparing the cell representativity of $k = 10$ and $k = 12$, which are 68.2% and 62.8%, respectively.

Convergence of the solutions Although the results obtained after 30h of execution for our selected parameters were promising, we extended our analysis further. We ran the $v2$ program for 20 days, with $k = 10$ and $l = 6$, monitoring the number of pseudo-perturbations identified by the program over time (Figure 5.5). As we stopped the program after 20 days, the pseudo-perturbations found are suboptimal solutions.

We observed an exponential increase in the number of identified pseudo-perturbations until a plateau was reached, with 96 identified pseudo-perturbations (consistent from day 7 to day 20). This convergence suggests that our results are probably close to optimality. Across time, we select 5 key points, identifying 10, 43, 78, 92 and 96 pseudo-perturbations. These key points are analyzed in detail in the following section.

A

k	l	#pseudo-perturbations
5	3	27
6	3	45
7	4	64
8	4	69
9	5	81
10	6	**92**
11	6	79
12	7	87
13	7	76
14	8	79
15	9	75

B

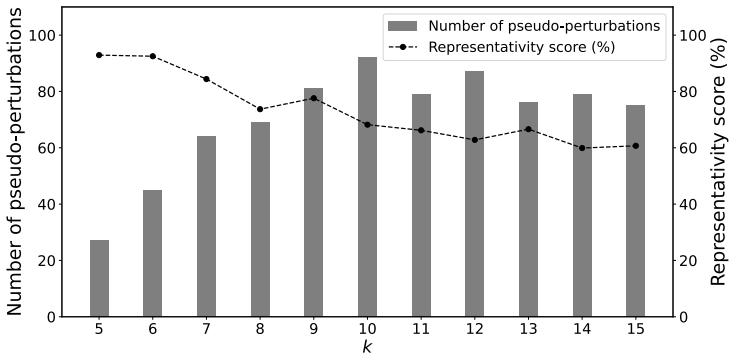


Figure 5.4 – Impact of k parameter.
The results were obtained after 30h of computation on a computer cluster.
(A) Identified pseudo-perturbations in function of k and l parameters.
(B) Identified pseudo-perturbations and the representativity score (see Equation 4.3) of these pseudo-perturbations for a set of k values.

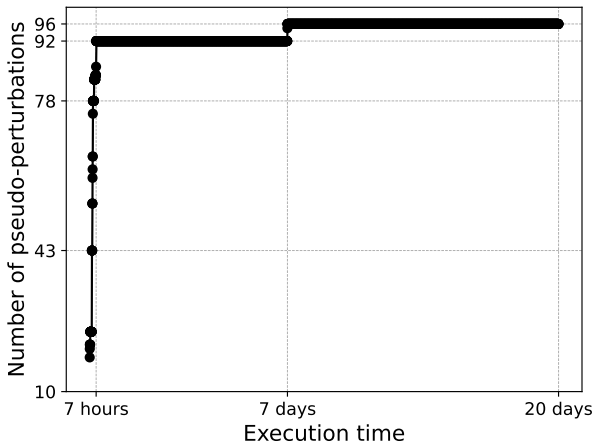


Figure 5.5 – Convergence of the identified pseudo-perturbations over time.
The results were obtained using a computer cluster, as the complexity of the calculations makes it infeasible to use a classical laptop.

Robustness of the solutions Recall that the ASP program identifies a given number of pseudo-perturbations for a specific set of k genes. However, other k-genes sets can yield the same number of pseudo-perturbations, which we refer to as *equivalent solutions*. We explored these equivalent solutions for each previously mentioned key points. To do this, we adapted the ASP program by fixing the number of pseudo-perturbation to identify and enumerate the possible (equivalent) solutions. We ran the enumeration for 7 days, stopping the program afterwards, yielding a (non-exhaustive) set of equivalent solutions

(Figure 5.6).

Observing Figure 5.6A, we identify a convergence in the number of equivalent solutions over time. Interestingly, there is a significant disparity in the number of solutions when comparing #pseudo-perturbations = 10 ($1,716,211$ solutions) to #pseudo-perturbations = 43 ($2,179,441$ solutions). One might expect the opposite results: a larger number of equivalent solution for the first key point (10). However, it is important to note that these numbers represent suboptimal values, as the calculations were halted after a predetermined 7-day timeout. For the 96 pseudo-perturbations plateau, 2 equivalent subsets of $k = 10$ genes were observed.

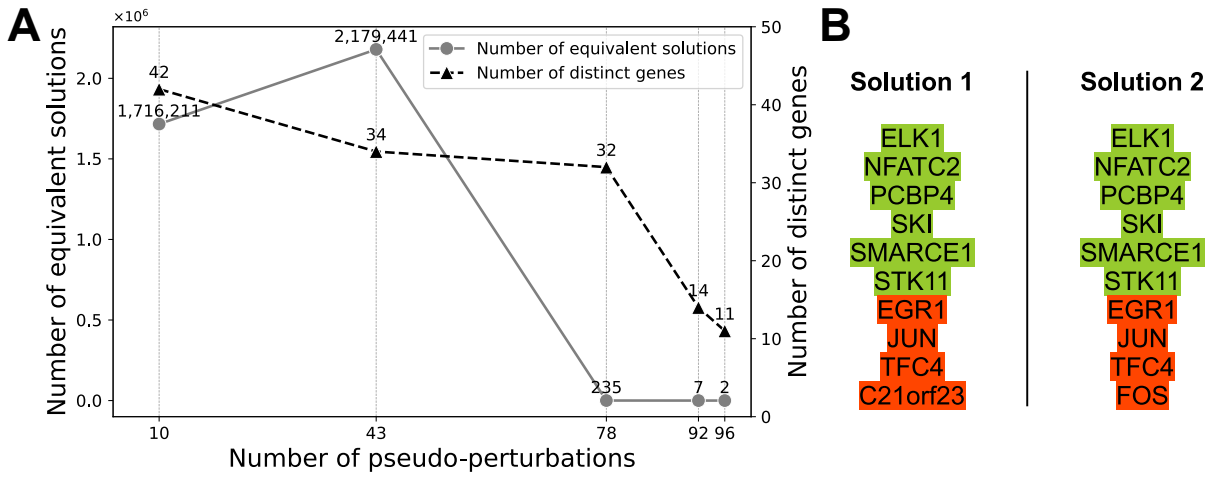


Figure 5.6 – Exploration of equivalent solutions and their gene composition.

This significant difference between the large quantity of equivalent solutions (more than 1 million) at the beginning of the pseudo-perturbation search and the few remaining solutions (only 2) could be explained by data sparsity. Due to the large number of zeros in the data, the probability of finding, for instance, 10 cells where a gene is not expressed (0) is high. In contrast, finding 96 cells (out of about 300) where a gene is not expressed is much less likely. Consequently, the number of equivalent solutions selecting different sets of genes could may grow exponentially for few pseudo-perturbations.

To address this issue, we implement a new constraint in the ASP program to forbid this phenomenon. However, this constraint significantly slowed down the solving process. Therefore, we decided not to include this constraint in the program.

This omission is not problematic because, for robust solutions (92 and 96 pseudo-

perturbations key points), we did not observe any genes always expressed at 0 in the pseudo-perturbations, indicating that our results are reliable.

In addition, we further analyzed the gene composition of these equivalent solutions, counting the number of distinct genes in the equivalent solutions (Figure 5.6A). The number of distinct genes decreased as the number of pseudo-perturbations increased. Among the more than 2 millions of solutions for the 43 pseudo-perturbations key point, only 34 different genes composed the subsets of $k = 10$ genes. For 96 pseudo-perturbations key point, 9 out of 10 genes were common between the two solutions, with only 1 gene differing: *C21orf23* and *FOS*, respectively (Figure 5.6B).

Both the equivalent solutions and gene composition exhibit a refinement as the number of pseudo-perturbations converges. The program found over 1 million equivalent solutions when analyzing 10 pseudo-perturbations key point, but this number reduced drastically to only 2 equivalent solutions when examining 96 pseudo-perturbations key point. These outcomes demonstrate the robustness of our solutions.

Cell representativity of the solutions We calculate the cell representativity score for the two solutions identifying 96 pseudo-perturbations (Table 5.3). Globally, the average representativity of the two developmental stages equals 76% (resp. 71%) for solution 1 (resp. solution 2). Additionally, these two average values are greater than the one obtained for 92 pseudo-perturbations during our k parameter exploration (68.2% ; see Section 5.3.3.1). The cell representativity can be used as a metric to select between equivalent solutions. Following this metric for the 96 pseudo-perturbations key point, the solution 1 should be selected. However, we consider both solutions for the remaining analyses, which are the readout difference maximization and the BN inference.

Table 5.3 – Cell representativity of solutions 1 and 2.

Solution	M^{TE} (%)	L^{TE} (%)	Total (%)
1	263 (76%)	253 (74%)	516 (76%)
2	248 (71%)	235 (75%)	483 (71%)

5.3.3.2 Computed experimental designs

As previously introduced, the data preprocessing step uses two readout normalizations: the “min-max” normalization and the “arctangeant” normalization. We consider these two

normalizations and compare them for the following analyses. Thus, for each normalization and given the two equivalent solutions comprising 96 pseudo-perturbations for two distinct sets of k genes, we maximize the readout differences following the process presented in previous chapter (see Section 4.2.5.2). This step allows us to form, for each equivalent solution, two experimental designs, each specific to the studied developmental stage. These experimental designs are composed of 6 inputs, 4 intermediates and 19 readouts. In Figure 5.7, we represent the experimental designs for the solution 1 to compare the two considered normalizations. We present 5 cells involved in the 96 pseudo-perturbations, where we describe the presence or absence of the inputs and intermediates, and plot the evolution between medium and late TE of the expression of 7 readout genes involved in the inferred BNs (see Section 5.3.4). In Figure 5.7A, we illustrate a part of the experimental design using "min-max" normalization, while in Figure 5.7B, we present the "arctangeant" normalization one. In these two partial experimental designs, the same genes and cells are used, illustrating the variation in the evolution of readout gene expression between the two normalization methods. We observe that more expression values are around middle values for "min-max", while "arctangeant" normalization values are closer to 0 or 1, consistent with the normalization distributions discussed in Section 5.2.1 (Figure 5.1). Similar trends are observed across the other cells among the 96 cells in pseudo-perturbations.

In addition, we conducted a similar analysis using solution 2 (see Appendix G). The same observations can be made leading to the similar differences between the two studied normalizations.

5.3.4 Inferred BNs

Considering both normalizations, we combined the reconstructed PKN (see Section 5.3.2) with the experimental designs constructed for both solutions containing 96 pseudo-perturbations (see Section 5.3.3.2). Using Caspo, specific families of BNs were inferred for the two solutions in order to model and distinguish medium and late TE developmental stages. We set the parameters to $fitness\_tolerance$ = 0.0001, $size\_tolerance$ = 0 and $length$ = 2, similar to the approach used in the previous chapter (see Section 4.3.3).

Comparison of normalizations In Table 5.4, we compare the BNs inferred for solutions 1 and 2 across both normalizations, showing the MSE and the size of the optimal learned BN, the number of learned BNs for both developmental stages. We also calculate

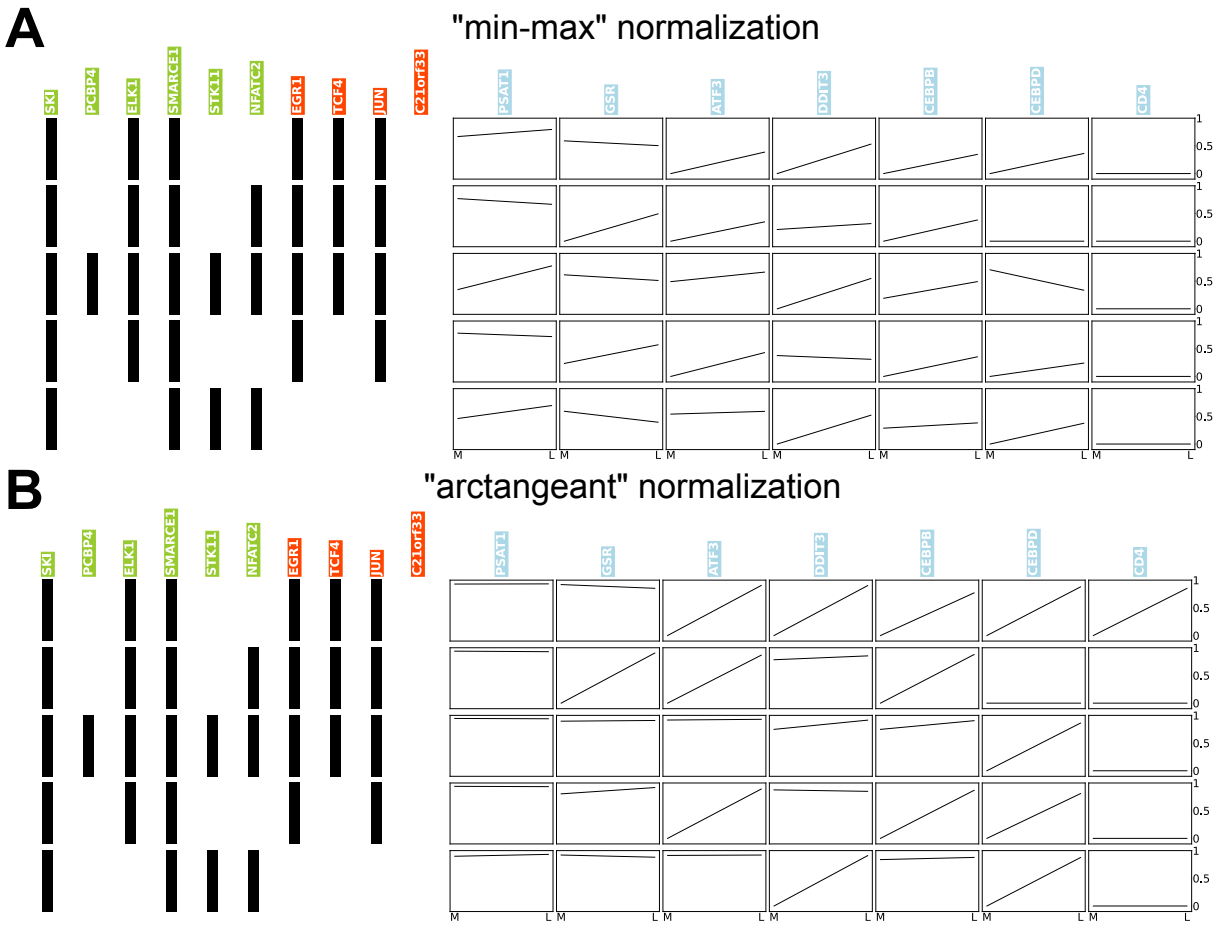


Figure 5.7 – Partial graphical representations of experimental designs discovered in solution 1 for both normalizations.

Only the 5 first cells involved in pseudo-perturbations are considered. Each row (left side) represents an optimal pseudo-perturbation on the 10 selected input (green) and intermediate (red) genes. Binarized vectors are illustrated using bars, where a black (resp. white) bar means the gene is active (resp. inactive). On the right side, readout genes in blue, present in the inferred BNs (cf. Figure 5.8), are shown. In each box, the curve represents the normalized readout gene expression evolution between the medium TE (M, left) and late TE (L, right) developmental stages.
(A) Representations for "min-max" normalization.
(B) Representations for "arctangeant" normalization.

the accuracy, obtained from the cell classifier method (see Section 5.2.5), on the 192 $(96 + 96)$ cells involved in pseudo-perturbations, for each stage.

The "min-max" normalization identifies a single medium TE BN for both solutions. The number of BNs increases significantly for late TE, with $1,496$ and 199 BNs for solutions 1 and 2, respectively. This larger BNs count for late TE indicates a greater complexity in regulatory mechanisms required to model this stage. Similarly, the size (i.e.,

Table 5.4 – Comparison of normalizations with solution 1 and 2 inferred BN characteristics.

Normalization	Solution	Medium TE				Late TE			
		MSE	Size	#BNs	Accuracy (training set)	MSE	Size	#BNs	Accuracy (training set)
"min-max"	1	0.1153	1	1	38%	0.1413	12	1,496	29%
	2	0.1180	1	1	52%	0.1400	5	199	11%
"arctangeant"	1	0.2218	23	1	64%	0.2416	15	2	81%
	2	0.2242	21	214	79%	0.2516	16	70	51%

The accuracies was calculated with cells from the cells involved in pseudo-perturbations used for the BN inference, i.e., learning set (see Section 5.3.5).

the number of Boolean gates in the network) is larger for late TE (12 and 5) compared to medium TE (1 for both solutions). For both solutions, gene expression observations are more complex to model for late TE, as indicated by the MSE, which quantifies the difference between data and model predictions. The inferred BN for medium TE (for both solutions) is particularly simple, consisting of only one link between two genes, making the model poor and less representative (see Figure 5.8B). In addition, the accuracies are very low, meaning an incorrect classification for a large majority of considered cells.

Given the "arctangeant" normalization, the second set of k genes (solution 2) allows the identification of a larger number of BNs than solution 1: 1 BN vs. 214 BNs and 2 BNs vs. 70 BNs for solution 1 and 2, respectively. We observe a similar trend for the MSE: late TE has a larger MSE than medium TE for both solutions. In contrast, the size of BNs is larger for medium TE. The calculated accuracies are high, going to a maximal value 81% for late TE, solution 1.

As far as MSE is concerned, the order of magnitude is the same for both solution, regardless of the normalization method. However, MSE increases for "arctangeant" normalization: 2 times higher for medium TE and approximately 1.7 times higher for late TE compared to the "min-max" normalization. A first explanation is the number of readout in inferred BNs and thus the values to fit. For instance in solution 1, the number of readout in "min-max" BNs is equal to 1, while for "arctangeant" BNs this number is 5. Fitting the values of 5 readouts is significantly more challenging than fitting 1 value, resulting in larger MSE. The second explanation involves incorrect predictions made through the BNs. In these cases, the distance between the observed value and the prediction will be smaller for "min-max" normalization, than for "arctangeant" one, leading to a smaller MSE for "min-max" normalization. For instance, if the prediction

through the BN is equal to 0 and the observed value is around 1 for the "arctangeant" normalization and around 0.5 for the "min-max" normalization, the distance for "min-max" normalization will be smaller, yielding a smaller MSE. These reasons could explain the larger MSE for "arctangeant" normalization.

Additionally, we observe that size values raise between medium to late TE for "min-max" normalization, while, a reduction is observed for "arctangeant" one. The number of BNs increases from medium to late for "min-max" normalization, but this trend is not as clear for the second normalization, and even reverses for solution 2 (214 vs. 70 for medium and late TE, respectively).

Furthermore, if we take cells used for the BN learning individually and compute the sorting using the cell classifier (see Section 5.2.5), we observe higher accuracies for "arctangeant" normalization (72% and 65%, for solutions 1 and 2, respectively) compared to the "min-max" normalization (33% and 32%, for solutions 1 and 2, respectively). These outcomes reinforce the robustness of "arctangeant" normalization. That is why we analyze in detail, in the following section, the inferred BNs obtained with "arctangeant" normalization.

Analysis of the inferred BNs obtained with "arctangeant" normalization As the cell representativity of the solution 1 is the greater (see Section 5.3.3.1), we analyze the BN families of this solution (Figure 5.8A). We can observe mostly genes involved in both BN families. More interactions are needed to model medium TE, as indicated by the larger size. These interactions are always necessary because Caspo learns only 1 BN for medium TE, while for late TE, alternative regulation mechanisms are possible, especially the inhibition of *DDIT3*, that can be mediated by *MYC* or through the *MYC/Max/MIZ-1* complex. Caspo models late TE with one additional readout (*CEBPD*) compared to medium TE. Additionally, we show the BNs obtained for "min-max" normalization in Figure 5.8B, reinforcing the better significance of "arctangeant" normalization results.

In summary, medium TE needs more diverse mechanisms to be modeled. For instance, an additional regulation cascade can be observed for *CEBPB*, involving a pathway from *JUN* and passing through *C21orf23* or *JUN/JUND*.

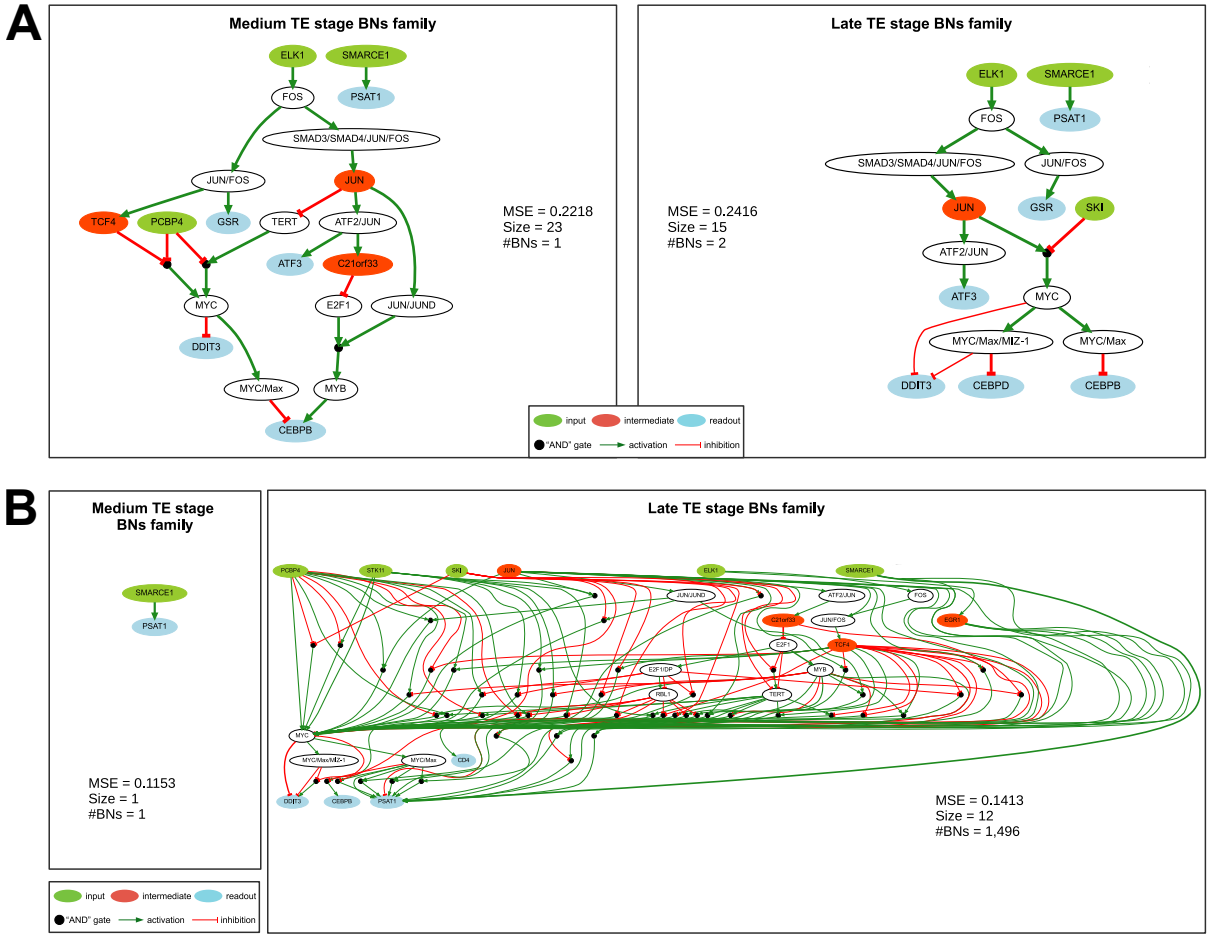


Figure 5.8 – Boolean networks families for solution 1.
Each network represents the union of (sub-)optimal Boolean networks (BNs) learned from the reduced PKN and the experimental design. The colored nodes represent genes associated with experimental designs, including input and intermediates involved in pseudo-perturbations, and readout genes. The width of the arcs represents the frequency of occurrence of this arc in the BNs. We set the following parameter values: $fitness\_tolerance$ = 0.0001, $size\_tolerance$ = 0 and $length$ = 2.
(A) BN families inferred for "arctangeant" normalization.
(B) BN families inferred for "min-max" normalization.

5.3.5 Cell classifier results

Here, we consider 2 sets of cells to classify: the "training set" and the "testing set". The first one contains the cells involved in pseudo-perturbations used for the BN learning. The testing set is divided in two parts. The first part, called "testing set 1", includes redundant cells exhibiting identical gene expression profiles to the pseudo-perturbation set. These cells are closer to the learning cells in terms of gene expression because they

have the same expression values for the input and intermediate genes. The second part, called "testing set 2", comprises the remaining cell population, excluding those involved in pseudo-perturbations or redundancies.

We employ our cell classifier on both solutions 1 and 2, utilizing both normalizations "min-max" and "arctangeant". The results are summarized in Table 5.5. A significant difference between the two normalizations is observed across all solutions, with "arctangeant" normalization demonstrating an approximate two-fold improvement. These findings reiterate the robustness of "arctangeant" normalization compared to "min-max", when using this data as input to Caspo.

Focusing solely on "arctangeant" normalization, the training set exhibits a higher BAC than the testing set, with values of 72% vs. 67%/68% for solution 1. Similar results are obtained for solution 2: 65% vs. 58%. These outcomes are consistent, as models tend to perform better with training that is more similar to the model itself, than testing data. However, this trend is not observed with "min-max" data.

Table 5.5 – Cell classifier results on solutions 1 and 2 considering the two normalizations.

Normalization	Solution	Training set				Testing set 1				Testing set 2			
		Global accuracy	BAC	Medium TE accuracy	Late TE accuracy	Global accuracy	BAC	Medium TE accuracy	Late TE accuracy	Global accuracy	BAC	Medium TE accuracy	Late TE accuracy
"min-max"	1	33%	33%	38%	29%	36%	35%	49%	22%	30%	30%	35%	24%
	2	32%	32%	52%	11%	33%	32%	55%	10%	29%	29%	45%	13%
"arctangeant"	1	72%	72%	64%	81%	66%	67%	52%	82%	68%	68%	62%	73%
	2	65%	65%	79%	51%	58%	58%	59%	57%	58%	58%	68%	47%

Based on this accuracy metric, solution 1 appears superior. This solution demonstrates higher accuracy, particularly for testing set. Notably, it achieves an approximate 10% improvement in accuracy for this solution. This suggests that the inferred BNs effectively model cells with significant differences compared to the training set.

We examine the predictions for each individual cell in solution 1 in more detail. In Figure 5.9, we represent the calculated MSEs through medium TE BNs versus the calculated MSEs through late TE BNs for each cell within the three sets of cells: learning set (Figure 5.9A), testing set 1 (B) and testing set 2 (C). For each set, we represent all cells across both classes (medium and late TE) in function of their medium and late TE MSEs in the left plot. The two plots on the right are subplots of the first one, considering cells from only one class. Given these plots, we can observe whether the classification of a cell is correct or not, based on its position on the plot. If the cell is below the dashed line,

it is classified in the medium TE group; otherwise, it is classified in the late TE group. Considering the testing set, we observe that most of misclassified medium TE cells are farther from the dashed line than the misclassified late TE ones (Figure 5.9A). The same observations can be made for the learning set 1 (Figure 5.9B). This signifies that medium TE BNs have more difficulties identifying medium TE cells. In contrast, for testing set 2, we observe a larger distance for misclassified cells in late TE compared to medium TE. This suggests that late TE struggle to identify cells that are far from the training data.

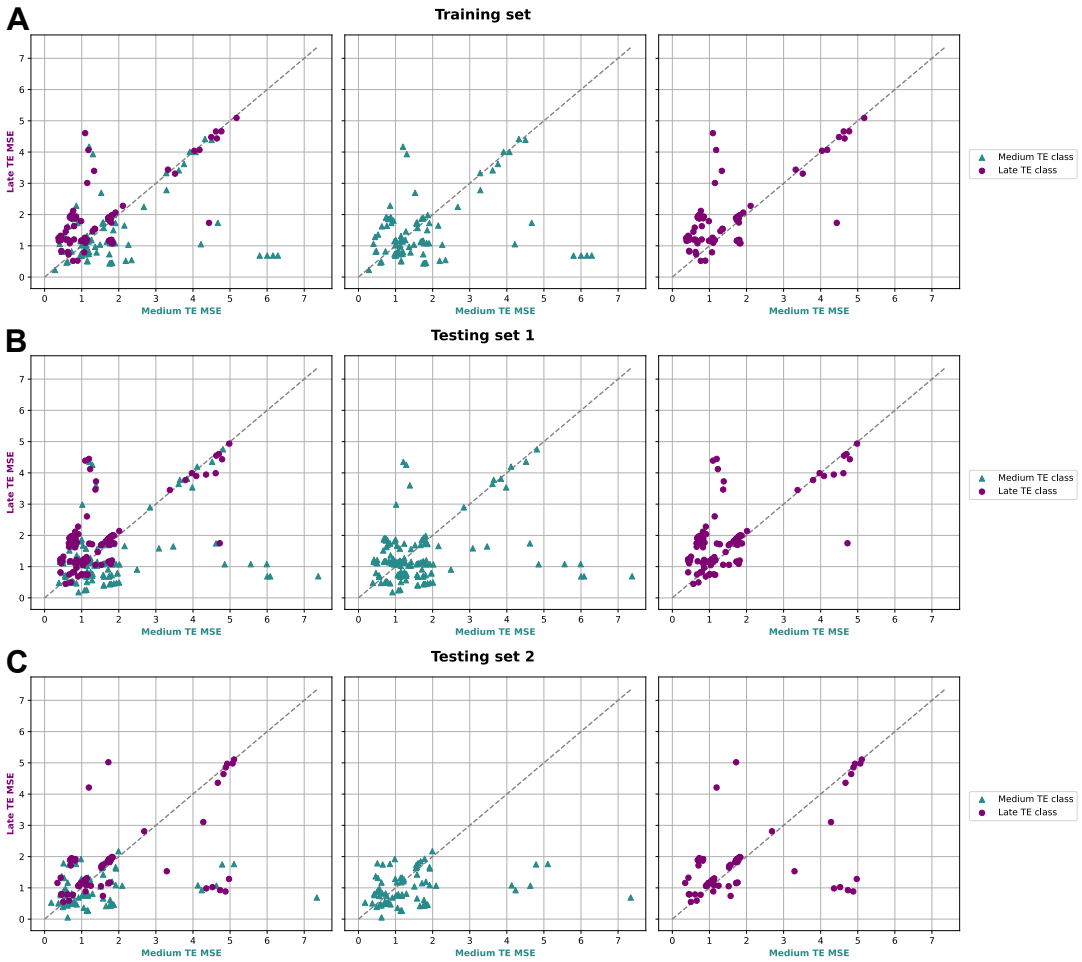


Figure 5.9 – MSE scores of individual cells for the solution 1.
Left plot represents the medium TE MSE in function of the late TE MSE. The two plots on the right are subplots of the left plot, considering cells from only one class. MSE values in negative logarithmic scale. A high value in this scale signifies a small MSE, meaning a good prediction.
(A) Results for training set (96 and 96 in medium and late TE class, respectively).
(B) Results for testing set 1 (167 and 96 in medium and late TE class, respectively).
(C) Results for testing set 2 (85 and 79 in medium and late TE class, respectively).

5.4 Discussion and conclusion

Understanding regulatory mechanisms involved in developmental stages and leading to cell fate decision is a complex research field, particularly in the study of human embryonic development. In Bolteau *et al.* [5] and in Chapter 4, we presented a method allowing us to identify pseudo-perturbations, since classical perturbations are unavailable due to legal and experimental concerns. Given these pseudo-perturbations, we inferred two families of Boolean networks (BNs), each modeling one of the studied developmental stages: medium trophectoderm (TE) and late TE. This method faces limitations such as computational constraints. To overcome these issues, we improved our previous method and introduced SCIBORG, which addresses the aforementioned issues and further analyzes our findings.

In SCIBORG we improved the $v1$ ASP program (from Chapter 4) for identifying pseudo-perturbations yielding better results while requiring less computation memory and time. This improvement allows us to consider larger input PKNs, permitting the incorporation of more prior knowledge. Additionally, we explored a readout gene expression normalization framework using an arctangeant function and compared it with the initial "min-max" normalization. We delved deeper into the identified pseudo-perturbations by exploring equivalent solutions for a certain number of identified pseudo-perturbations. We also introduced a cell classifier aimed at sorting a given cell into the developmental stage that is closest to the cell's gene observations. Compared to our previous outcomes, in this study, we worked with a larger prior knowledge base, comprising 42 additional genes and 84 additional interactions. The improved $v2$ ASP program enabled the identification of 92 pseudo-perturbations for 7 hours of computation, compared with previously 20 pseudo-perturbations found in 65 hours. Thus, the $v2$ ASP program increased the number of pseudo-perturbations by a factor of 4.6 while reducing the computation time by more than 50 hours. We identified a convergence of the number of pseudo-perturbations until a plateau comprising 96 pseudo-perturbations. This number of 96 pseudo-perturbations demonstrates robustness since *(i)* two equivalent solutions are found, *(ii)* the composition of genes forming these solutions comprises 9 (out of 10) identical genes, and *(iii)* the cell representativity of pseudo-perturbations is approximately 70%. The new "arctangeant" normalization enables the learning of more informative BNs compared to "min-max" until then used. Finally, the cell classifier provides interesting results and forms a tool for classification of new cells. The inferred BNs exhibit more complex regulations mechanisms for medium TE than for late TE (size 23 vs 15). More

genes seem to be involved in the medium TE stage.

Although we applied SCIBORG to distinguish medium TE and late TE, we consider that our method could be applied to other developmental stages. Our methodology could also be used to understand regulatory mechanisms in other biological fields. Other PKNs could be used as input; if a PKN already exists in a particular domain, this should only improve the regulatory mechanisms identified in the BNs. Furthermore, our method is complementary to modeling approaches using single-cell transcriptomic data and model cell differentiation [61, 66]. Our method considers all cells of the population and their regulatory mechanisms for studied developmental stages. SCIBORG infers robust and exhaustive BNs, providing insightful inputs for biologists.

Chapter 6

Discussion and Perspectives

"Science knows no country, because knowledge belongs to humanity, and is the torch which illuminates the world."

— Louis Pasteur (1822-1895)

Summary

This chapter provides an overall discussion of the work conducted during this thesis. It concludes by outlining the future perspectives of this research.

6.1 Discussion

In this manuscript, we presented two contributions leaded in this thesis. The first one introduces a method of Boolean network (BN) inference aiming to model two stages of human embryonic development. In the second contribution, we outline enhancements leading to exhaustive enumeration and optimal families of models, allowing us to distinguish medium and late trophectoderm (TE) developmental stages. The assembly

of these two contributions led to SCIBORG, an original tool that allows to highlight the main mechanisms involved in the human embryonic development. SCIBORG is available as a Python package, which can be accessed through the following GitHub repository: TODO.

SCIBORG consists of three steps: *(i)* prior knowledge network (PKN) reconstruction, *(ii)* experimental design construction, and *(iii)* BN inference. First, the PKN reconstruction calls a tool which queries the Pathway Commons database in order to reconstruct a gene interaction graph constituting our prior knowledge. Second, to construct stage-specific experimental designs, we first use an ASP program to identify pseudo-perturbations. Then, we aim to maximize the readout differences across cells selected by the pseudo-perturbations. Third, given the PKN and experimental designs, we use a tool encoded in ASP to infer BNs that are both compatible with the gene interactions into the PKN, and the gene expression present in experimental designs.

Applied to medium and late TE stages, SCIBORG allows us to identify a PKN comprising 233 genes and protein-complexes, and 369 gene interactions. Our pseudo-perturbation identification program enables the identification of 96 pseudo-perturbations in both studied stages. Exploring the pseudo-perturbation equivalent solutions, we found two solutions composed of two subsets of 10 genes allowing the identification of 96 pseudo-perturbations; that is, 96 cells in each line which expression profile allowed us to differentiate medium and late TE stages. We consider two readouts normalization methodologies and identify that "arctangeant" normalization enables the learning of more robust and pertinent results. In the third step, we learned two families of BNs modeling medium and late TE. We identify different regulatory mechanisms, specific to the studied stages. Furthermore, our inferred BNs permit the introduction of a cell classifier, aiming to sort a cell into the closer stage, giving its gene expression.

Ultimately, the findings of the second contribution, presented in Chapter 5, will be compiled into a scientific article and submitted to an international journal for publication.

6.1.1 Data preprocessing

In the preprocessing step, the data is either binarized, for input and intermediate genes, or normalized for readout genes.

We use a simple binarization method, considering a gene to be expressed if at least two reads are observed in the cell. However, this approach is open to discussion. Exploring different threshold values could be interesting to see if they impact the binarization results.

In addition, other binarization methods exist, such as PROFILE [8] used in BoNesis, or RefBool [9]. These methods, however, may assign intermediate expression levels to certain genes, or discards some genes. Applying these methods as they are could potentially result in excluding certain genes or cells (where a gene is not considered), which cannot be managed yet in SCIBORG.

Regarding the readout normalization, the methodology used significantly impacts the inferred BNs (enhancement of two fold increase in accuracy when comparing "arctangent" normalization to "min-max" normalization.). The "arctangeant" normalization provides more robust and pertinent BNs. Nonetheless, other normalization methods could be explored further to potentially yield better results.

6.1.2 PKN reconstruction

Our PKN reconstruction method leverages pyBRAvo which queries Pathway Commons, a resource grouping multiple databases. Pathway Commons includes KEGG [10], the database used in the study by Chebouba *et al.* [11], which allows us to incorporate more prior knowledge than that used in their study. We opted to use an external database to reconstruct the PKN instead of a PKN inference method using scRNAseq data, such as LEAP (see Chapter 2, Section 2.4), because the PKN inferred from these methods may be biased by the data.

pyBRAvo proposes to reconstruct two types of graphs: gene regulation networks (GRNs) or signaling networks (SNs). In SCIBORG, we use the regulation option to focus on GRNs. However, considering the signaling option could be beneficial. Even though SNs contain signaling information, treating them as PKNs might yield different and potentially interesting results.

Furthermore, our reconstructed PKNs include numerous gene interactions derived from cancer analysis data. However, the mechanisms involved in cancer differ from those in embryonic development, a concern that needs to be addressed. A potential solution could be using the DoRothEA database [12], as employed in BoNesis. This database contains interactions with varying confidence levels, from high confidence (curated interactions) to low confidence (predictions), thus necessitating cautious use. Another option is to use an inference method based on scRNAseq data, always mindful of the potential data bias in the PKN.

Lastly, we reconstruct two different PKNs (PKN^0 and PKN^2), which vary in size. It was observed that the larger PKN, PKN^0, yielded more relevant inferred BNs (see

Chapter 5, Section 5.3.4). Despite the increase in size, SCIBORG was able to handle this problem effectively.

6.1.3 Usage of other PKNs

It is important to note that the PKN reconstruction step is optional in SCIBORG. Users possessing a PKN tailored to their specific case study have the flexibility to utilize it and execute the subsequent method steps. Our approach aims to be as adaptable as possible to accommodate different user-specific PKNs.

6.1.4 Improving readout maximization step

The readout difference maximization is implemented in Python and takes place after the pseudo-perturbation identification in ASP. The algorithm is straightforward, calculating pairwise differences in cell redundancies (see Chapter 4, Section 4.2.5.2). While one could consider enhancing this process with other algorithms, it was not a priority since the execution time is reasonably short (less than 5 minutes on a laptop).

Another option would be to incorporate additional constraints into the pseudo-perturbation identification ASP program, in order to maximize readout differences directly in the ASP solving process. However, this approach would be very resource-intensive, significantly complicating and lengthening the search for pseudo-perturbations.

Furthermore, we employ a maximization of the readout differences to handle cell redundancies, aiming to achieve the largest difference between the two studied stages and potentially better distinguish them. However, other criteria could be considered. We explored an "average readout" value criterion, which involves computing the average readout values of all redundant cells. The learned BNs obtained with this criterion were not significantly different from those obtained with the difference maximization approach, leading us to retain the latter in SCIBORG.

6.1.5 Caspo's parameters

To infer Boolean networks through Caspo, we used various parameters.

The first parameter considered is the fitness tolerance, which aims to provide tolerance in terms of MSE for learning BNs. In our two contributions, we fix the fitness tolerance value to $fitness\_tolerance$ = 0.0001, allowing exploration beyond the optimal BN up

to a distance of 0.01% from the optimal MSE. We chose this value because is the one providing in average the more pertinent BNs.

The second parameter is the size tolerance, which sets a tolerance level to explore BNs with a number of nodes ranging from the optimal size up to a user-defined maximum tolerance added to the optimal number of nodes. We opted not to provide tolerance in terms of size because our goal is to find minimal Boolean models.

The third parameter is the length, which restricts the number of incoming interactions in an "AND" logical gate. We set this length to 2 in order to simplify regulatory mechanisms being inferred, thereby reducing problem complexity.

Exploring these parameters more deeply by testing various values could potentially yield more robust outcomes.

6.1.6 Improvement of inferred BNs

The composition of inferred BNs are sometimes unsatisfactory. In our first contribution, the learned BNs included intermediates directly connected to readouts (see Chapter 4, Section 4.3.3), which is not biologically informative. To address this issue, we incorporated additional constraints in the pseudo-perturbation identification ASP program ($v2$) of SCIBORG, in order to select inputs and intermediates that are connected (see Chapter 5, Section 5.2.3, *Constraint 4*). This approach enabled the inference of more relevant BNs.

However, the learned BNs remain disconnected when using "arctangeant" normalization (see Chapter 5, Section 5.3.4). To infer more connected BNs, which include "cascades" from inputs to readouts via intermediates, the ASP program of Caspo needs to be modified at its core. This task is challenging as it requires a thorough understanding of each ASP rule to make the necessary modifications, but it may lead to more meaningful results, which is invaluable for modeling.

6.1.7 State-of-the-art methods

The modeling methods using SMT or ASP enumerate a massive quantity of possible models. State-of-the-art methods address this explosion of solutions by sampling models (BoNesis or RE:IN), or relying on experiments. In contrast, SCIBORG enables the identification of equivalent models over time (see Chapter 5, Section 5.3.3.1). We demonstrate that the number of possible models is very limited. Compared to state-

of-the-art methods, our solution space is more restricted, with only 2 models versus some hundreds or thousands. These outcomes provide a high level of confidence in our results and strongly support the directed experimental validation of our models.

Furthermore, we observe that SCIBORG considers cell heterogeneity through the identification of several cells (around 100) within each class. For each cell, we consider a screen of approximately 30 genes, which expression profile enables us to learn the BNs. In contrast, BoNesis works with an average of cells within a class, while RE:IN uses bulk data. SCNS considers single-cell data over a window of approximately 40 genes; however, it does not handle redundant expression across cells. Accounting for cell heterogeneity enables SCIBORG to consider the redundancies commonly present in single-cell transcriptomic data, a factor that is not taken into account in other state-of-the-art methods.

6.2 Perspectives

The work led during this thesis is a significant step forward in modeling human embryonic development. Although our research focused on the modeling of medium and late trophectoderm (TE), there remain numerous areas to be explored.

Firstly, we plan to analyze other developmental stages involved in the human preimplantation development. The immediate goal is to continue the exploration of TE maturation by including early TE, a phase occurring before the others. Following this, we aim to extend our analysis to other developmental stages to investigate the other cell fates: primitive endoderme (PrE) and epiblast (EPI).

Secondly, we plan a deeper investigation of the cell classifier proposed with SCIBORG to sort cells into their corresponding stages. This tool could assist in classifying undefined cells exhibited in the analyses conducted by Meistermann *et al.* [13] (see Chapter 3, Section 3.2). To achieve this, we need to explore neighboring stages of these cells, particularly focusing on EPI and PrE, which ties into the previous point discussed.

The inferred BNs could be challenged through simulation, such as deactivating a gene and observing the resulting behavior of the BNs and gene expression. This powerful approach would facilitate in silico predictions, providing valuable insights and potentially guiding experimental validation.

In parallel with the computational models we infer, cellular models, particularly

blastoids (see Chapter 1), are promising tools for computation models validation with less legal constraints. Blastoids could be very useful for providing in vitro validation of in silico simulations and thus confirming posed hypotheses.

Furthermore, we plan to apply SCIBORG on other biological studies. We are collaborating with a group at the CRCI2NA laboratory in Nantes, who are working on inner lymphoid cell development. Their aim is to understand gene regulatory mechanisms involved in this cell differentiation. SCIBORG could be instrumental in identifying unknown mechanisms in this process.

Ultimately, SCIBORG facilitates the static modeling through the inferred BNs. An intriguing direction for future work is the modeling of the dynamic processes occurring during the embryonic development. For this purpose, an extension of Caspo, known as Caspo Time Series (Caspo-ts) [14], could be employed. Given a prior knowledge network (PKN) and time series data, which could be inferred from pseudotime gene expression in our case, Caspo-ts infers BNs that are compatible with both gene interactions defined in the PKN, and the gene expression patterns identified in time series data.

Preliminary work has been conducted by students I co-supervised with Carito Guziolowski, exploring this approach and yielding promising findings. This perspective forms the basis of a new Ph.D. thesis set to start shortly. I am very happy to see that the work conducted during my thesis will be continued and expanded upon.

Appendices

Appendix A

Technical specification sheet of SCENIC

SCENIC (single-cell regulatory network inference and clustering) [51] is a workflow designed to infer GRNs and identify cell states from scRNAseq data. This method combines gene interaction data with transcription factor (TF) binding motif enrichment, enabling the discovery and characterization of cell states.

SCENIC consists of three steps (Figure A.1):

1. **Co-expression identification:** Similar to WGCNA, SCENIC identifies set of co-expressed genes using the GENIE3 algorithm [102]. From gene expression and a list of TFs of interest, the algorithm identifies *modules* which are sets of TFs co-expressed with their targets (Figure A.1a). However, like WGCNA, this genes modules, based only on co-expression, may include many false positives.
2. **TF-motif enrichment analysis:** The RcisTarget program is used to analyze TF-motif enrichment and identify direct targets. This step retains only modules where the TF motif is enriched, forming what the authors called *regulons* (Figure A.1b). Together, modules and regulons are used to form the GRN.
3. **Regulon activity calculation:** The activity of each regulon is calculated for each cell using the AUCell algorithm (Figure A.1c). The calculated scores can be represented as a regulon activity matrix. By applying a threshold, it is possible to determine in which cells each regulon is "on". Additionally, the regulon activity matrix can be used to cluster cell and potentially identify cell types (Figure A.1d).

In summary, SCENIC models pairwise relationships between genes using co-expression correlations. Gene correlations are filtered using a TF enrichment to retain only TFs and their predicted targets in the final GRN. The method also proposes cell type-specific GRNs by exploring the regulon activity. By using TF binding motifs from database, SCENIC can infer directed GRNs, showing the impact of TFs on their targets. In theory, repressive

interaction can be detected via the enrichment; however, this remains challenging for certain data [51].

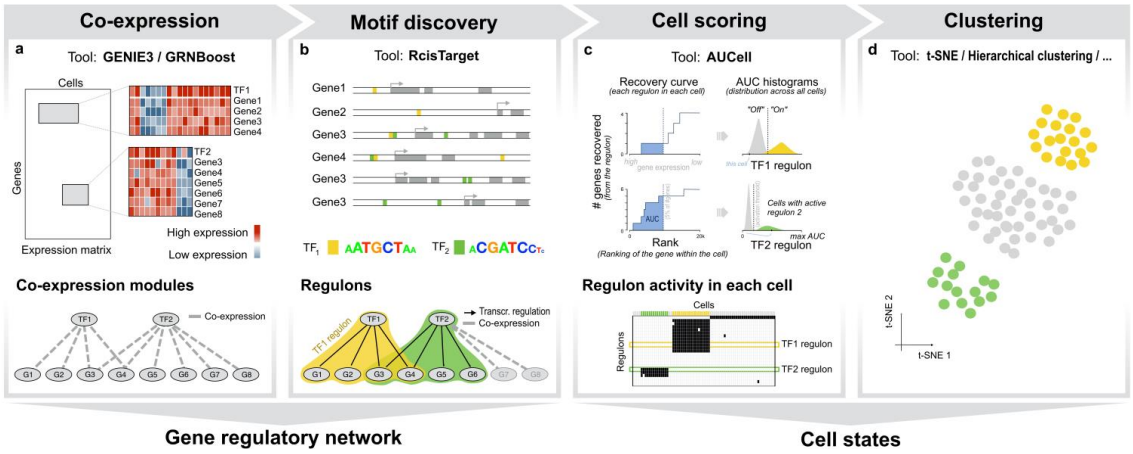


Figure A.1 – The SCENIC workflow.

(a) GENIE3 is used to infer co-expression *modules* between transcription factors (TFs) and candidate target genes.
(b) Each co-expressed module is analyzed using RcisTarget to identify enriched TF-motifs. Only modules and targets for which the TF-motif is enriched are retained, forming together *regulons*.
(c) The AUCell algorithm calculates the activity of each regulon for each cell, forming a regulon activity matrix. This matrix can be binarized using a threshold in order to, for each regulon, determine in which cells the regulon is "on".
(d) The regulon activity matrix can be utilized to cluster cell (e.g., t-SNE) and potentially identify cell types.
Figure from Aibar et al. *[51].*

Appendix B

Technical specification sheet of LEAP

LEAP (lag-based expression association for pseudotime-series) [55] is a method for inferring gene co-expression network from pseudotime trajectories. It calculates the co-expression while accounting for potential lags in time present in scRNAseq data, acknowledging that gene relationships may vary over time. Overall, the method consists of two main steps (Figure B.1):

1. **Pseudotime trajectories calculation:** Pseudotime trajectories are calculated to order cells. LEAP requires the user to provide the time ordering of cells, for instance, using Monocle (Figure B.1i).
2. **Gene correlation calculation:** LEAP divides the data in small time windows representing all possible time lags. For each possible time lag, gene correlations are computed, resulting in a series of correlation matrices (Figure B.1ii). These multiple correlations matrices are then merged using Pearson correlation, enabling the inference of temporary correlations between genes. Finally, multiple correlations are aggregated into an adjacency matrix indicating the sign of correlation between genes (Figure B.1iii). This matrix can easily be transformed into a GRN.

In summary, LEAP allows researchers to capture pairwise relationships between genes using scRNAseq, constructing a GRN that accounts for hidden associations due to time lags. The pseudotime trajectories enable the computation of non-symmetric correlation, providing the ability to predict directed GRNs. The method's output can help in understanding complex processes involved in cell fate decisions for instance.

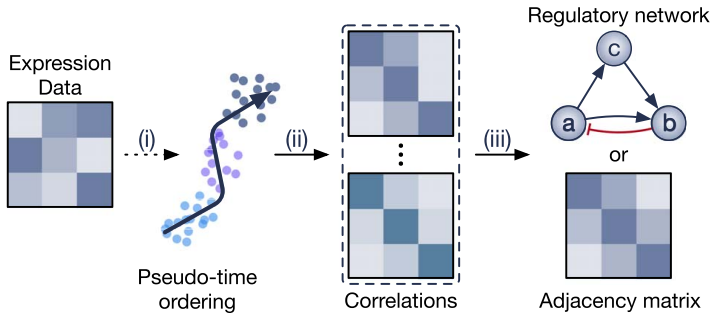


Figure B.1 – Workflow of GRN inference method based on pseudotime ordering (LEAP).
(i) The pseudotime trajectories are calculated from scRNAseq, providing an ordering of cells.
(ii) Data is divided into small-time windows, and the algorithm calculates the gene correlation for each time window.
(iii) The method merges multiple correlation matrices into one adjacency matrix, which can be then transformed into a GRN.
Figure from Nguyen et al. *[50].*

Appendix C

Technical specification sheet of WGCNA

WGCNA (weighted gene co-expression network analysis) [71] is a widely used approach for exploring relationships between genes with similar expression patterns. Its goal is to identify pairwise correlations across transcriptomic data to find co-expressed genes grouped into *modules*. The output is a gene co-expression network where nodes represent genes, and edges represent pairwise correlations between gene expressions. Briefly, the method comprises three main steps (Figure C.1):

1. **Correlation matrix construction:** A correlation matrix is created from gene expression data using a similarity measure such as Pearson correlation. The algorithm calculates correlation between every pair of genes. This output matrix can be represented as an unsigned graph, with nodes as genes and edges weighted by correlation intensity (Figure C.1a).
2. **Thresholding:** A threshold is applied to retain high correlations and remove weak connections, resulting in a simplified network of strong gene connections (Figure C.1b).
3. **Module identification:** The algorithm identifies gene modules using a network proximity measure. A distance matrix between genes is calculated to identify clusters, which form the gene modules (Figure C.1c). Within each module a *hub* gene can be identified, corresponding to a gene highly connected to others module genes.

In summary, WGCNA is an unsupervised method that identifies gene modules from the expression data using clustering techniques. These modules can be enriched by gene ontology information to identify potential biological pathways for instance. Additionally, hub genes within modules can serve as candidate biomarkers. Various studies have identified potential biomarkers using WGCNA [103, 104]. However, the lack of causal

regulatory links proposed by this method can lead to a high number of false positives [49].

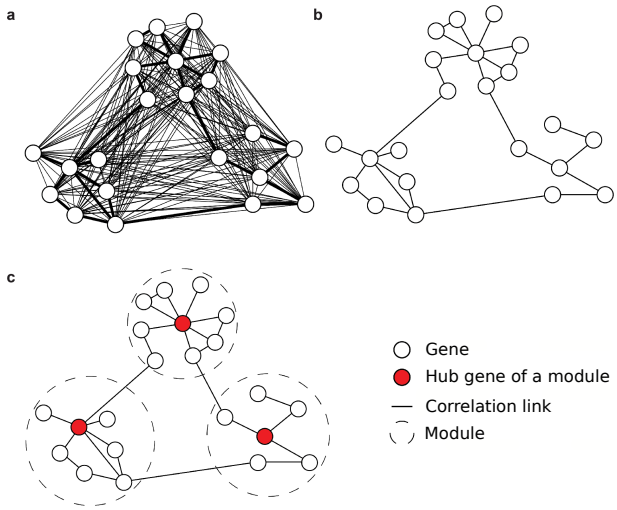


Figure C.1 – Schematic diagram of the WGCNA algorithm.
(a) The correlation matrix can be represented with an unsigned graph where nodes are genes. The correlation intensity is indicated by the weight of the edge between two genes.
(b) Correlation values below a threshold are removed, simplifying the graph to include only strong connections.
(c) The graph is partitioned to identify gene modules composed of highly correlated genes. For each module, the algorithm highlights a *hub* gene (in red), which is a gene highly connected with the other genes in the module.
Figure adapted from Meistermann [4].

Appendix D

Transcription factors list

Below we list the transcription factors used as input of pyBRAvo tool for the PKN reconstruction (see Section 4.3.3, PKN reconstruction).

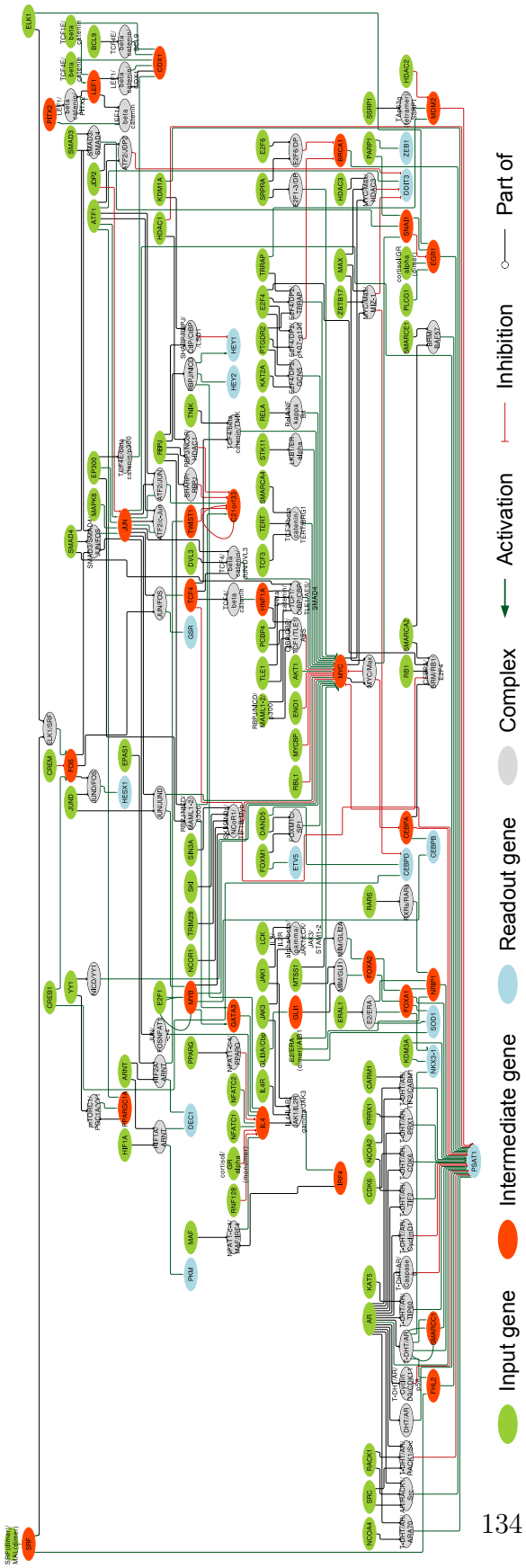
ACO1	DAB2	FHL2	HIST1H2BN	KLF4	NKX6-1	PRDM11
AKR1A1	DBP	FIGLA	HIST2H2BE	KLF6	NKX6-2	PRDM14
ANXA1	DDIT3	FLI1	HKR1	KLF7	NMI	PRDM16
ANXA11	DDX4	FOSB	HLF	KLF9	NNT	PRDX5
ARG2	DDX43	FOSL1	HNF1A	KLRG1	NOBOX	PRKAA1
ARID5B	DLX1	FOXA2	HNF4A	LARP1	NR1H4	PRKAA2
ASH2L	DLX2	FOXD2	HOXA4	LEF1	NR2E1	PSMC2
ATF2	DLX3	FOXN2	HOXA7	LHX2	NR2F2	PSMD12
BACH2	DLX4	FOXO3	HOXA9	LHX5	NR3C1	RAB14
BARHL2	DLX5	FOXP1	HOXB13	LHX8	NR3C2	RAB18
BARX1	DMRTB1	FOXQ1	HOXB6	LRRFIP1	NR4A3	RARB
BARX2	DMRTC2	GATA2	HOXC10	LSM6	NR5A2	RAX2
BATF	DNMT1	GATA3	HOXD8	LUZP2	NR6A1	RBBP9
BCL11A	DPRX	GATA4	HTATIP2	MAF	NRF1	RELB
BCL3	DUXA	GBX1	HUNK	MCTP2	OLIG1	RFX4
BHLHE40	E2F8	GBX2	ING3	MECOM	OSR1	RFXANK
CARF	EBF1	GCM1	IRF3	MIXL1	OSR2	RLF
CBFA2T2	EBF2	GIT2	IRF4	MLXIPL	OTX1	RORB
CCDC25	EBF3	GOT1	IRF6	MSI2	OTX2	RUNX1
CDX1	ECSIT	GPD1	IRF8	MSRA	OVOL1	RUNX2
CDX2	EGR1	GRHL1	IRX2	MSRB3	OVOL2	RUVBL1
CEBPA	EGR2	GRHL2	IRX3	MTHFD1	P4HB	RXRA
CEBPB	EIF5A2	GRHL3	IRX4	MYCL	PARP1	SALL2
CEBPD	ELF3	GRHPR	IRX5	MYLK	PAX9	SATB1
CELF5	ELK3	GTF2A1L	ISL2	NANOG	PBX3	SETBP1
CERS2	EMX1	GTF3A	JAZF1	NANOGP8	PIR	SHOX2
CERS3	EN1	H2AFY	JDP2	NCALD	PITX2	SIN3A
CERS6	ERG	HAND1	JRKL	NEUROG2	PKM	SMAD5
CLOCK	ESRRG	HCFC2	KDM4A	NFATC1	PKNOX2	SMAD6
CPEB1	ETFB	HES1	KDM4D	NFE2	PLAG1	SNAI1
CREB5	ETS2	HESX1	KDM4E	NFIX	PLAGL1	SNAI3
CREBL2	ETV1	HEY1	KLF11	NFKB1	POLD2	SND1
CTBP1	ETV4	HEY2	KLF12	NFKB2	POU5F1B	SOD1
CTCFL	ETV5	HHAT	KLF17	NFYA	PPARG	SOX1
CTNNB1	EXO5	HHEX	KLF18	NKX2-5	PPARGC1A	SOX11
CUX2	EZR	HIF1A	KLF2	NKX3-1	PRDM1	SOX15
CYB5R1	FEZF2	HIRIP3	KLF3	NKX3-2	PRDM10	SOX17

SOX2	TFCP2L1	ZFP3	ZNF174	ZNF394	ZNF599	ZNF727
SOX30	TFEB	ZFP37	ZNF18	ZNF408	ZNF606	ZNF735
SOX4	TGIF2LX	ZFP42	ZNF182	ZNF416	ZNF610	ZNF736
SOX5	THAP1	ZFP62	ZNF184	ZNF438	ZNF616	ZNF766
SOX9	THRB	ZFP64	ZNF19	ZNF439	ZNF630	ZNF79
SP110	TIGD2	ZFP90	ZNF200	ZNF440	ZNF654	ZNF814
SP6	TOPORS	ZHX2	ZNF211	ZNF479	ZNF668	ZNF829
SPIC	TP63	ZHX3	ZNF214	ZNF490	ZNF669	ZNF830
SSX3	TPI1	ZIC3	ZNF215	ZNF506	ZNF674	ZNF831
STAT3	TPPP	ZIK1	ZNF226	ZNF528	ZNF675	ZNF844
STAT5A	TRIB2	ZIM2	ZNF23	ZNF530	ZNF677	ZNF845
SUCLG1	TRIB3	ZIM3	ZNF230	ZNF534	ZNF679	ZNF878
TAF7	TRIP10	ZKSCAN4	ZNF25	ZNF541	ZNF684	ZNF880
TAGLN2	TULP1	ZKSCAN5	ZNF256	ZNF549	ZNF689	ZNF891
TBPL2	UBE2V1	ZNF10	ZNF266	ZNF555	ZNF69	ZNF92
TBX2	UGP2	ZNF117	ZNF280A	ZNF557	ZNF697	ZRSR2
TBX3	VENTX	ZNF132	ZNF284	ZNF559	ZNF701	ZSCAN10
TBX5	YWHAZ	ZNF134	ZNF304	ZNF561	ZNF702P	ZSCAN18
TCF24	YY2	ZNF136	ZNF329	ZNF569	ZNF705A	ZSCAN32
TCF7L1	ZBTB11	ZNF140	ZNF331	ZNF574	ZNF705CP	ZSCAN4
TCF7L2	ZBTB16	ZNF146	ZNF341	ZNF578	ZNF705D	ZSCAN5A
TEAD1	ZBTB49	ZNF155	ZNF343	ZNF584	ZNF705G	ZSCAN5B
TEAD3	ZBTB7B	ZNF157	ZNF350	ZNF586	ZNF706	ZSCAN5C
TFAP2A	ZCCHC14	ZNF16	ZNF354A	ZNF595	ZNF708	
TFAP2B	ZEB1	ZNF165	ZNF362	ZNF596	ZNF714	
TFAP2D	ZFHX3	ZNF17	ZNF385A	ZNF597	ZNF716	

Appendix E

Reconstructed Prior Knowledge Network PKN^2

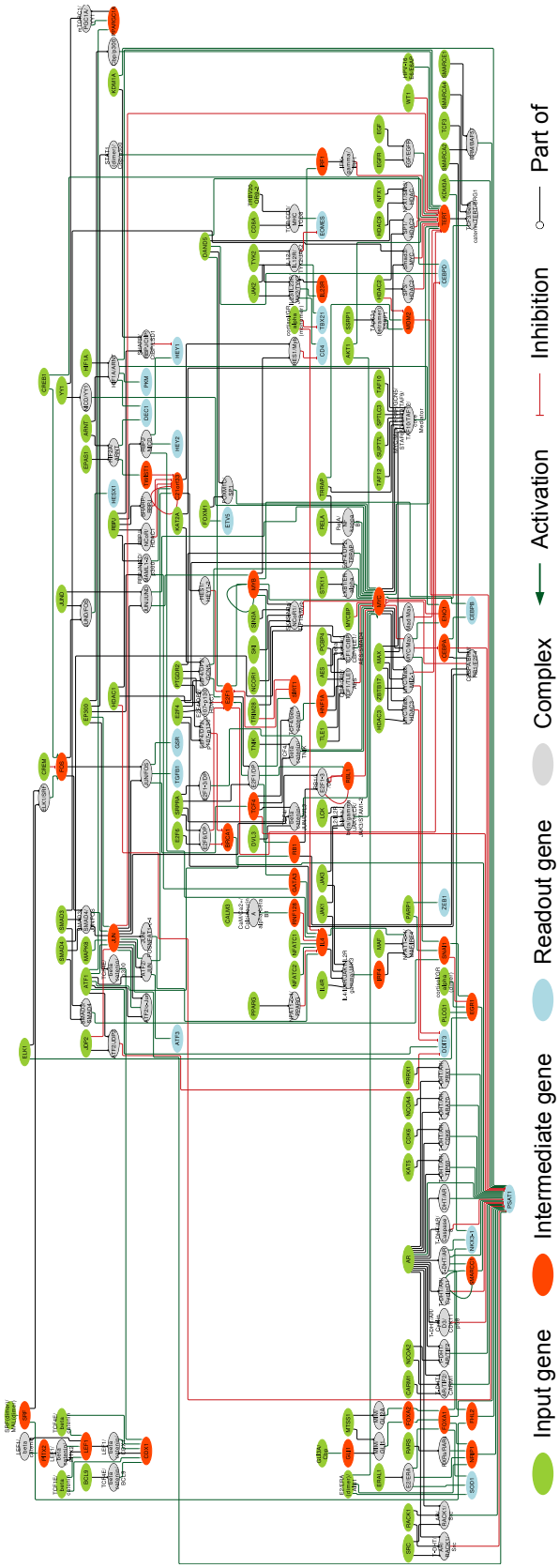
See next page.


Input gene
Intermediate gene
Readout gene
Complex
Activation
Inhibition
Part of

Appendix F

Reconstructed Prior Knowledge Network PKN^0

See next page.


Input gene
Intermediate gene
Readout gene
Complex
Activation
Inhibition
Part of

Experimental designs of solution 2

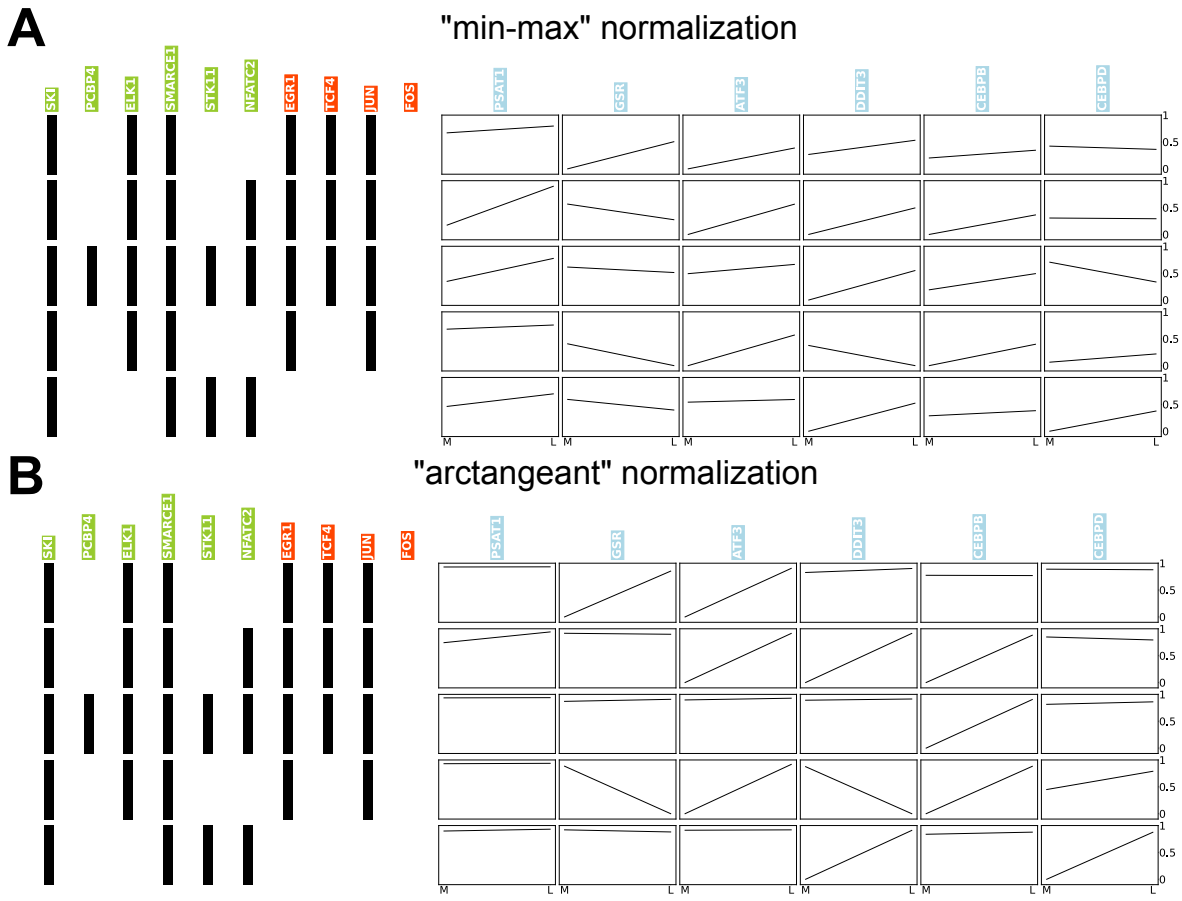


Figure G.1 – Partial graphical representations of experimental designs discovered in solution 2 for both normalizations.

Only the 5 first cells involved in pseudo-perturbations are considered. Each row (left side) represents an optimal pseudo-perturbation on the 10 selected input (green) and intermediate (red) genes. Binarized vectors are illustrated using bars, where a black (resp. white) bar means the gene is active (resp. inactive). On the right side, readout genes in blue, present in the inferred BNs, are shown. In each box, the curve represents the normalized readout gene expression evolution between the medium TE (M, left) and late TE (L, right) developmental stages.
(A) Representations for “min-max” normalization.
(B) Representations for “arctangeant” normalization.

BIBLIOGRAPHY

Author's Publications

5. Bolteau, M., Bourdon, J., David, L. & Guziolowski, C., *Inferring Boolean Networks from Single-Cell Human Embryo Datasets* en, *in Bioinformatics Research and Applications* (eds Guo, X., Mangul, S., Patterson, M. & Zelikovsky, A.) (Springer Nature, Singapore, 2023), 431–441, ISBN: 9789819970742 (cit. on pp. xiv, xviii, 6, 9, 59, 87, 113).

6. Bolteau, M., Chebouba, L., David, L., Bourdon, J. & Guziolowski, C., Boolean Network Models of Human Preimplantation Development, *Journal of Computational Biology,* `https://www.liebertpub.com/doi/abs/10.1089/cmb.2024.0517` (2024) (May 2024) (cit. on pp. xvi, xviii, 8 sq., 59).

7. Le Bars, S., Bolteau, M., Bourdon, J. & Guziolowski, C., Predicting weighted unobserved nodes in a regulatory network using answer set programming, en, *BMC Bioinformatics* **24,** 321, ISSN: 1471-2105, `https://doi.org/10.1186/s12859-023-05429-3` (2024) (Aug. 2023) (cit. on pp. xvii, 8, 40).

Other Publications

1. Wolpert, L., Tickle, C. & Arias, A. M., *Principles of Development* en, Google-Books-ID: WbO6BwAAQBAJ, ISBN: 9780198709886 (Oxford University Press, 2015) (cit. on pp. ix, 1).

2. De Geyter, C., Calhaz-Jorge, C., Kupka, M. S., Wyns, C., Mocanu, E., Motrenko, T., Scaravelli, G., Smeenk, J., Vidakovic, S., Goossens, V. & The European IVF-monitoring Consortium (EIM) for the European Society of Human Reproduction and Embryology (ESHRE), ART in Europe, 2014: results generated from European registries by ESHRE†: The European IVF-monitoring Consortium (EIM)‡ for the European Society of Human Reproduction and Embryology (ESHRE), *Human*

Reproduction **33,** 1586–1601, ISSN: 0268-1161, https://doi.org/10.1093/humrep/dey242 (2024) (Sept. 2018) (cit. on pp. ix, 2).

3. Kagawa, H., Javali, A., Khoei, H. H., Sommer, T. M., Sestini, G., Novatchkova, M., Scholte op Reimer, Y., Castel, G., Bruneau, A., Maenhoudt, N., Lammers, J., Loubersac, S., Freour, T., Vankelecom, H., David, L. & Rivron, N., Human blastoids model blastocyst development and implantation, en, *Nature* **601,** 600–605, ISSN: 1476-4687, https://www.nature.com/articles/s41586-021-04267-8 (2024) (Jan. 2022) (cit. on pp. x, 2 sq.).

4. Meistermann, D., *Modélisation du développement préimplantatoire humain à partir de données de transcriptome de cellule unique* 2020NANT1019, PhD thesis (2020), http://www.theses.fr/2020NANT1019/document (cit. on pp. xii, 4, 18, 130).

8. Béal, J., Montagud, A., Traynard, P., Barillot, E. & Calzone, L., Personalization of Logical Models With Multi-Omics Data Allows Clinical Stratification of Patients, *Frontiers in Physiology* **9,** ISSN: 1664-042X, https://www.frontiersin.org/articles/10.3389/fphys.2018.01965 (Jan. 2019) (cit. on pp. xix, 25, 117).

9. Jung, S., Hartmann, A. & del Sol, A., RefBool: a reference-based algorithm for discretizing gene expression data, *Bioinformatics* **33,** 1953–1962, ISSN: 1367-4803, https://doi.org/10.1093/bioinformatics/btx111 (2023) (July 2017) (cit. on pp. xix, 117).

10. Kanehisa, M. & Goto, S., KEGG: Kyoto Encyclopedia of Genes and Genomes, *Nucleic Acids Research* **28,** 27–30, ISSN: 0305-1048, https://doi.org/10.1093/nar/28.1.27 (2024) (Jan. 2000) (cit. on pp. xx, 55, 117).

11. Chebouba, L., Miannay, B., Boughaci, D. & Guziolowski, C., Discriminate the response of Acute Myeloid Leukemia patients to treatment by using proteomics data and Answer Set Programming, *BMC Bioinformatics* **19,** 15–26 (2018) (cit. on pp. xx, 55 sqq., 64, 73, 76, 78, 83, 117).

12. Garcia-Alonso, L., Holland, C. H., Ibrahim, M. M., Turei, D. & Saez-Rodriguez, J., Benchmark and integration of resources for the estimation of human transcription factor activities, en, *Genome Research* **29,** 1363–1375, ISSN: 1088-9051, 1549-5469, https://genome.cshlp.org/content/29/8/1363 (2024) (Jan. 2019) (cit. on pp. xx, 117).

13. Meistermann, D., Bruneau, A., Loubersac, S., Reignier, A., Firmin, J., François-Campion, V., Kilens, S., Lelièvre, Y., Lammers, J., Feyeux, M., Hulin, P., Nedellec, S., Bretin, B., Castel, G., Allègre, N., Covin, S., Bihouée, A., Soumillon, M., Mikkelsen, T., Barrière, P., Chazaud, C., Chappell, J., Pasque, V., Bourdon, J., Fréour, T. & David, L., Integrated pseudotime analysis of human pre-implantation embryo single-cell transcriptomes reveals the dynamics of lineage specification, *Cell Stem Cell* **28,** 1625–1640.e6 (2021) (cit. on pp. xxiv, 14 sq., 18, 37, 39 sq., 120).

14. Razzaq, M., Paulevé, L., Siegel, A., Saez-Rodriguez, J., Bourdon, J. & Guziolowski, C., Computational discovery of dynamic cell line specific Boolean networks from multiplex time-course data, *PLOS Computational Biology* **14** (ed Stelling, J.) e1006538–e1006538, ISSN: 1111111111, `https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006538` (Oct. 2018) (cit. on pp. xxiv, 121).

15. David, L., Bruneau, A., Fréour, T., Rivron, N. & Van de Velde, H., An update on human pre- and peri-implantation development: a blueprint for blastoids, *Current Opinion in Genetics & Development* **83,** 102125, ISSN: 0959-437X, `https://www.sciencedirect.com/science/article/pii/S0959437X23001053` (2024) (Dec. 2023) (cit. on pp. x, 2).

16. Burgaud, M., Bretin, B., Reignier, A., Vos, J. D. & David, L., Du nouveau dans les modèles d'étude de l'embryon humain, fr, *médecine/sciences* **39,** 129–136, ISSN: 0767-0974, 1958-5381, `https://www.medecinesciences.org/articles/medsci/abs/2023/02/msc220234/msc220234.html` (2024) (Feb. 2023) (cit. on pp. x, 2).

17. Kitano, H., Computational systems biology, en, *Nature* **420,** 206–210, ISSN: 1476-4687, `https://www.nature.com/articles/nature01254` (2024) (Nov. 2002) (cit. on p. 12).

18. Machado, D., Costa, R. S., Rocha, M., Ferreira, E. C., Tidor, B. & Rocha, I., Modeling formalisms in Systems Biology, en, *AMB Express* **1,** 45, ISSN: 2191-0855, `https://doi.org/10.1186/2191-0855-1-45` (2024) (Dec. 2011) (cit. on p. 12).

19. Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* en, Google-Books-ID: AvNID7LyMusC, ISBN: 9781558604797 (Morgan Kaufmann, Sept. 1988) (cit. on p. 12).

20. Petri, C. A., Kommunikation mit Automaten, ger, *http://edoc.sub.uni-hamburg.de/informatik/volltexte/2011/160/pdf/diss_petri.pdf,* https://edoc.sub.uni-hamburg.de/informatik/volltexte/2011/160/ (2024) (1962) (cit. on p. 12).

21. Kauffman, S., Metabolic stability and epigenesis in randomly constructed genetic nets, *Journal of Theoretical Biology* **22,** 437–467, ISSN: 0022-5193 (1969) (cit. on p. 12).

22. Wang, R.-S., Saadatpour, A. & Albert, R., Boolean modeling in systems biology: an overview of methodology and applications, en, *Physical Biology* **9,** 055001, ISSN: 1478-3975, https://dx.doi.org/10.1088/1478-3975/9/5/055001 (2024) (Sept. 2012) (cit. on pp. 12, 30).

23. Villaverde, A. F. & Banga, J. R., Reverse engineering and identification in systems biology: strategies, perspectives and challenges, *Journal of The Royal Society Interface* **11,** 20130505, https://royalsocietypublishing.org/doi/full/10.1098/rsif.2013.0505 (2024) (Feb. 2014) (cit. on p. 12).

24. Lowe, R., Shirley, N., Bleackley, M., Dolan, S. & Shafee, T., Transcriptomics technologies, *PLoS Computational Biology* **13,** e1005457, ISSN: 1553-734X, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5436640/ (2024) (May 2017) (cit. on p. 13).

25. Wang, Z., Gerstein, M. & Snyder, M., RNA-Seq: a revolutionary tool for transcriptomics, *Nature reviews. Genetics* **10,** 57–63, ISSN: 1471-0056, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2949280/ (2024) (Jan. 2009) (cit. on p. 13).

27. Guennewig, B., Lim, J., Marshall, L., McCorkindale, A. N., Paasila, P. J., Patrick, E., Kril, J. J., Halliday, G. M., Cooper, A. A. & Sutherland, G. T., Defining early changes in Alzheimer's disease from RNA sequencing of brain regions differentially affected by pathology, en, *Scientific Reports* **11,** 4865, ISSN: 2045-2322, https://www.nature.com/articles/s41598-021-83872-z (2024) (Mar. 2021) (cit. on p. 13).

28. Tang, F., Barbacioru, C., Wang, Y., Nordman, E., Lee, C., Xu, N., Wang, X., Bodeau, J., Tuch, B. B., Siddiqui, A., Lao, K. & Surani, M. A., mRNA-Seq whole-transcriptome analysis of a single cell, en, *Nature Methods* **6,** 377–382, ISSN: 1548-

7105, https://www.nature.com/articles/nmeth.1315 (2024) (May 2009) (cit. on p. 13).

29. Jovic, D., Liang, X., Zeng, H., Lin, L., Xu, F. & Luo, Y., Single-cell RNA sequencing technologies and applications: A brief overview, en, *Clinical and Translational Medicine* **12,** e694, ISSN: 2001-1326, https://onlinelibrary.wiley.com/doi/abs/10.1002/ctm2.694 (2024) (2022) (cit. on p. 14).

30. Olsen, T. K. & Baryawno, N., Introduction to Single-Cell RNA Sequencing, eng, *Current Protocols in Molecular Biology* **122,** e57, ISSN: 1934-3647 (Apr. 2018) (cit. on p. 14).

31. Hwang, B., Lee, J. H. & Bang, D., Single-cell RNA sequencing technologies and bioinformatics pipelines, en, *Experimental & Molecular Medicine* **50,** 1–14, ISSN: 2092-6413, https://www.nature.com/articles/s12276-018-0071-8 (2024) (Aug. 2018) (cit. on p. 14).

32. Chen, G., Ning, B. & Shi, T., Single-cell RNA-seq technologies and related computational data analysis, *Frontiers in genetics* **10,** 441123, ISSN: 1664-8021 (2019) (cit. on p. 14).

33. Kolodziejczyk, A. A., Kim, J. K., Svensson, V., Marioni, J. C. & Teichmann, S. A., The Technology and Biology of Single-Cell RNA Sequencing, *Molecular Cell* **58,** 610–620, ISSN: 1097-2765, https://www.sciencedirect.com/science/article/pii/S1097276515002610 (2024) (May 2015) (cit. on p. 14).

34. Van den Berge, K., Perraudeau, F., Soneson, C., Love, M. I., Risso, D., Vert, J.-P., Robinson, M. D., Dudoit, S. & Clement, L., Observation weights unlock bulk RNA-seq tools for zero inflation and single-cell applications, en, *Genome Biology* **19,** 24, ISSN: 1474-760X, https://doi.org/10.1186/s13059-018-1406-4 (2024) (Feb. 2018) (cit. on p. 15).

35. Hicks, S. C., Townes, F. W., Teng, M. & Irizarry, R. A., Missing data and technical variability in single-cell RNA-sequencing experiments, eng, *Biostatistics (Oxford, England)* **19,** 562–578, ISSN: 1468-4357 (Oct. 2018) (cit. on p. 15).

36. Jiang, R., Sun, T., Song, D. & Li, J. J., Statistics or biology: the zero-inflation controversy about scRNA-seq data, *Genome Biology* **23,** 31, ISSN: 1474-760X (2022) (cit. on p. 15).

37. Hotelling, H., Analysis of a complex of statistical variables into principal components, *Journal of Educational Psychology* **24,** 417–441, ISSN: 1939-2176 (1933) (cit. on p. 16).

38. Van der Maaten, L. & Hinton, G., Visualizing data using t-SNE. *Journal of machine learning research* **9** (2008) (cit. on pp. 16 sq.).

39. McInnes, L., Healy, J., Saul, N. & Großberger, L., UMAP: Uniform Manifold Approximation and Projection, *Journal of Open Source Software* **3,** 861 (2018) (cit. on pp. 16 sq., 38).

40. Sainburg, T., McInnes, L. & Gentner, T. Q., Parametric UMAP Embeddings for Representation and Semisupervised Learning, *Neural Computation* **33,** 2881–2907, ISSN: 0899-7667, https://doi.org/10.1162/neco_a_01434 (2024) (Oct. 2021) (cit. on p. 17).

41. Becht, E., McInnes, L., Healy, J., Dutertre, C.-A., Kwok, I. W. H., Ng, L. G., Ginhoux, F. & Newell, E. W., Dimensionality reduction for visualizing single-cell data using UMAP, en, *Nature Biotechnology* **37,** 38–44, ISSN: 1546-1696, https://www.nature.com/articles/nbt.4314 (2024) (Jan. 2019) (cit. on p. 17).

42. Xiang, R., Wang, W., Yang, L., Wang, S., Xu, C. & Chen, X., A Comparison for Dimensionality Reduction Methods of Single-Cell RNA-seq Data, *Frontiers in Genetics* **12,** ISSN: 1664-8021, https://www.frontiersin.org/journals/genetics/articles/10.3389/fgene.2021.646936 (2021) (cit. on p. 17).

43. Cannoodt, R., Saelens, W. & Saeys, Y., Computational methods for trajectory inference from single-cell transcriptomics, *European Journal of Immunology* **46,** 2496–2506, ISSN: 1521-4141, https://onlinelibrary.wiley.com/doi/abs/10.1002/eji.201646347 (2024) (2016) (cit. on p. 18).

44. Chen, H., Albergante, L., Hsu, J. Y., Lareau, C. A., Lo Bosco, G., Guan, J., Zhou, S., Gorban, A. N., Bauer, D. E., Aryee, M. J., Langenau, D. M., Zinovyev, A., Buenrostro, J. D., Yuan, G.-C. & Pinello, L., Single-cell trajectories reconstruction, exploration and mapping of omics data with STREAM, en, *Nature Communications* **10,** 1903, ISSN: 2041-1723, https://www.nature.com/articles/s41467-019-09670-4 (2024) (Apr. 2019) (cit. on pp. 18, 25).

47. Saelens, W., Cannoodt, R., Todorov, H. & Saeys, Y., A comparison of single-cell trajectory inference methods, en, *Nature Biotechnology* **37,** 547–554, ISSN: 1546-1696, https://www.nature.com/articles/s41587-019-0071-9 (2024) (May 2019) (cit. on p. 18).

48. Qiu, X., Hill, A., Packer, J., Lin, D., Ma, Y. A. & Trapnell, C., Single-cell mRNA quantification and differential analysis with Census, *Nat Methods* **14,** 309–315 (Jan. 2017) (cit. on pp. 18 sq., 38).

49. Badia-i-Mompel, P., Wessels, L., Müller-Dott, S., Trimbour, R., Ramirez Flores, R. O., Argelaguet, R. & Saez-Rodriguez, J., Gene regulatory network inference in the era of single-cell multi-omics, en, *Nature Reviews Genetics* **24,** 739–754, ISSN: 1471-0064, https://www.nature.com/articles/s41576-023-00618-5 (2024) (Nov. 2023) (cit. on pp. 19, 130).

50. Nguyen, H., Tran, D., Tran, B., Pehlivan, B. & Nguyen, T., A comprehensive survey of regulatory network inference methods using single cell RNA sequencing data, *Briefings in Bioinformatics* **22,** bbaa190, ISSN: 1477-4054, https://doi.org/10.1093/bib/bbaa190 (2024) (May 2021) (cit. on pp. 19–22, 128).

51. Aibar, S., González-Blas, C. B., Moerman, T., Huynh-Thu, V. A., Imrichova, H., Hulselmans, G., Rambow, F., Marine, J.-C., Geurts, P., Aerts, J., *et al.,* SCENIC: single-cell regulatory network inference and clustering, *Nature methods* **14,** 1083–1086 (2017) (cit. on pp. 20, 125 sq.).

52. Guo, M., Wang, H., Potter, S. S., Whitsett, J. A. & Xu, Y., SINCERA: A Pipeline for Single-Cell RNA-Seq Profiling Analysis, en, *PLOS Computational Biology* **11,** e1004575, ISSN: 1553-7358, https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004575 (2024) (Nov. 2015) (cit. on p. 20).

53. Liu, H., Li, P., Zhu, M., Wang, X., Lu, J. & Yu, T., Nonlinear Network Reconstruction from Gene Expression Data Using Marginal Dependencies Measured by DCOL, en, *PLOS ONE* **11,** e0158247, ISSN: 1932-6203, https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0158247 (2024) (July 2016) (cit. on p. 20).

54. Chan, T. E., Stumpf, M. P. H. & Babtie, A. C., Gene Regulatory Network Inference from Single-Cell Data Using Multivariate Information Measures, English, *Cell Systems* **5,** 251–267.e3, ISSN: 2405-4712, 2405-4720, https://www.cell.com/cell-systems/abstract/S2405-4712(17)30386-1 (2024) (Sept. 2017) (cit. on p. 20).

55. Specht, A. T. & Li, J., LEAP: constructing gene co-expression networks for single-cell RNA-sequencing data using pseudotime ordering, *Bioinformatics (Oxford, England)* **33,** 764–766, https://pubmed.ncbi.nlm.nih.gov/27993778/ (2017) (cit. on pp. 21, 127).

56. Cordero, P. & Stuart, J. M., *in*, 576–587 (WORLD SCIENTIFIC, Nov. 2016), ISBN: 9789813207806, https://www.worldscientific.com/doi/abs/10.1142/9789813207813_0053 (2024) (cit. on p. 21).

57. Papili Gao, N., Ud-Dean, S. M. M., Gandrillon, O. & Gunawan, R., SINCERITIES: inferring gene regulatory networks from time-stamped single cell transcriptional expression profiles, *Bioinformatics* **34,** 258–266, ISSN: 1367-4803, https://doi.org/10.1093/bioinformatics/btx575 (2024) (Jan. 2018) (cit. on p. 21).

58. Matsumoto, H., Kiryu, H., Furusawa, C., Ko, M. S. H., Ko, S. B. H., Gouda, N., Hayashi, T. & Nikaido, I., SCODE: an efficient regulatory network inference algorithm from single-cell RNA-Seq during differentiation, *Bioinformatics* **33,** 2314–2321, ISSN: 1367-4803, https://doi.org/10.1093/bioinformatics/btx194 (2024) (Aug. 2017) (cit. on p. 22).

59. Matsumoto, H. & Kiryu, H., SCOUP: a probabilistic model based on the Ornstein–Uhlenbeck process to analyze single-cell expression data during differentiation, en, *BMC Bioinformatics* **17,** 232, ISSN: 1471-2105, https://doi.org/10.1186/s12859-016-1109-3 (2024) (June 2016) (cit. on p. 22).

60. Ocone, A., Haghverdi, L., Mueller, N. S. & Theis, F. J., Reconstructing gene regulatory dynamics from high-dimensional single-cell snapshot data, *Bioinformatics* **31,** i89–i96, ISSN: 1367-4803, https://doi.org/10.1093/bioinformatics/btv257 (2024) (June 2015) (cit. on p. 22).

61. Chevalier, S., Froidevaux, C., Pauleve, L. & Zinovyev, A., Synthesis of boolean networks from biological dynamical constraints using answer-set programming, *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI* **2019-November,** 34–41, ISSN: 9781728137988 (2019) (cit. on pp. 23, 30, 114).

62. Chevalier, S., Noël, V., Calzone, L., Zinovyev, A. & Paulevé, L., *Synthesis and Simulation of Ensembles of Boolean Networks for Cell Fate Decision* en, *in Computational Methods in Systems Biology* (eds Abate, A., Petrov, T. & Wolf, V.) (Springer International Publishing, Cham, 2020), 193–209, ISBN: 9783030603274 (cit. on pp. 23, 83, 86).

63. Chevalier, S., *Inférence logique de réseaux booléens à partir de connaissances et d'observations de processus de différenciation cellulaire* 2022UPASG061, PhD thesis (2022), http://www.theses.fr/2022UPASG061/document (cit. on pp. 25, 30).

64. Nestorowa, S., Hamey, F. K., Pijuan Sala, B., Diamanti, E., Shepherd, M., Laurenti, E., Wilson, N. K., Kent, D. G. & Göttgens, B., A single-cell resolution map of mouse hematopoietic stem and progenitor cell differentiation, *Blood* **128,** e20–e31, ISSN: 0006-4971, https://doi.org/10.1182/blood-2016-05-716480 (2024) (Aug. 2016) (cit. on p. 25).

65. Garcia-Alonso, L., Holland, C. H., Ibrahim, M. M., Turei, D. & Saez-Rodriguez, J., Benchmark and integration of resources for the estimation of human transcription factor activities, *Genome research* **29,** 1363–1375 (2019) (cit. on p. 25).

66. Dunn, S.-J., Martello, G., Yordanov, B., Emmott, S. & Smith, A. G., Defining an essential transcription factor program for naïve pluripotency, *Science* **344,** 1156–1160, https://www.science.org/doi/10.1126/science.1248882 (2024) (June 2014) (cit. on pp. 26, 30, 114).

67. Yordanov, B., Dunn, S.-J., Kugler, H., Smith, A., Martello, G. & Emmott, S., A method to identify and analyze biological programs through automated reasoning, en, *npj Systems Biology and Applications* **2,** 1–16, ISSN: 2056-7189, https://www.nature.com/articles/npjsba201610 (2024) (July 2016) (cit. on pp. 26 sq.).

68. Dunn, S. J., Li, M. A., Carbognin, E., Smith, A. & Martello, G., A common molecular logic determines embryonic stem cell self-renewal and reprogramming, *EMBO J* **38** (Jan. 2019) (cit. on pp. 26, 30, 83, 86).

69. Moignard, V., Woodhouse, S., Haghverdi, L., Lilly, A. J., Tanaka, Y., Wilkinson, A. C., Buettner, F., Macaulay, I. C., Jawaid, W., Diamanti, E., Nishikawa, S.-I., Piterman, N., Kouskoff, V., Theis, F. J., Fisher, J. & Göttgens, B., Decoding the regulatory network of early blood development from single-cell gene expression measurements, en, *Nature Biotechnology* **33,** 269–276, ISSN: 1546-1696, https://www.nature.com/articles/nbt.3154 (2024) (Mar. 2015) (cit. on pp. 27–31, 87).

70. Petropoulos, S., Edsgärd, D., Reinius, B., Deng, Q., Panula, S. P., Codeluppi, S., Reyes, A. P., Linnarsson, S., Sandberg, R. & Lanner, F., Single-cell RNA-seq reveals lineage and X chromosome dynamics in human preimplantation embryos, *Cell* **165,** 1012–1026 (2016) (cit. on p. 37).

71. Langfelder, P. & Horvath, S., WGCNA: An R package for weighted correlation network analysis, English, *BMC Bioinformatics* **9,** ISSN: 1471-2105 (2008) (cit. on pp. 38, 77, 129).

72. Kowalski, R., *Predicate logic as programming language in IFIP congress* **74** (1974), 569–544 (cit. on p. 40).

73. Apt, K. R., Logic Programming. *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)* **1990,** 493–574 (1990) (cit. on p. 40).

74. Baral, C., *Knowledge Representation, Reasoning, and Declarative Problem Solving* ISBN: 0521818028 (Cambridge University Press, New York, NY, USA, 2003) (cit. on p. 40).

75. Lifschitz, V., *Answer set programming* (Springer Heidelberg, 2019) (cit. on p. 40).

76. Lifschitz, V., Answer set programming and plan generation, *Artificial Intelligence* **138,** 39–54 (2002) (cit. on p. 40).

77. Nogueira, M., Balduccini, M., Gelfond, M., Watson, R. & Barry, M., An a-prolog decision support system for the space shuttle, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **1990,** 169–183, `https://scholars.ttu.edu/en/publications/an-a-prolog-decision-support-system-for-the-space-shuttle` (2001) (cit. on p. 40).

78. Boenn, G., Brain, M., Vos, M. D. & Ffitch, J., Automatic music composition using answer set programming, *Theory and Practice of Logic Programming* **11,** 397–427, ISSN: 1471-0684, `https://researchportal.bath.ac.uk/en/publications/automatic-music-composition-using-answer-set-programming` (Mar. 2011) (cit. on p. 40).

79. Brooks, D. R., Erdem, E., Erdoğan, S. T., Minett, J. W. & Ringe, D., Inferring phylogenetic trees using answer set programming, *Journal of Automated Reasoning* **39,** 471–511 (2007) (cit. on p. 40).

80. Durzinsky, M., Marwan, W., Ostrowski, M., Schaub, T. & Wagler, A., Automatic network reconstruction using ASP, *Theory and Practice of Logic Programming* **11,** 749–766 (2011) (cit. on p. 40).

81. Schaub, T. & Thiele, S., Metabolic Network Expansion with Answer Set Programming, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **5649 LNCS,** 312–326, https://link.springer.com/chapter/10.1007/978-3-642-02846-5_27 (2009) (cit. on p. 40).

82. Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T. & Schneider, M., Potassco: The Potsdam Answer Set Solving Collection, *AI Commun.* **24,** 107–124 (Jan. 2011) (cit. on p. 44).

83. Gebser, M., Kaminski, R., Kaufmann, B. & Schaub, T., Answer Set Solving in Practice, *Synthesis Lectures on Artificial Intelligence and Machine Learning* **19,** 1–240 (Dec. 2012) (cit. on p. 44).

84. Gebser, M., Kaminski, R., Kaufmann, B. & Schaub, T., Answer set solving in practice, *Synthesis lectures on artificial intelligence and machine learning* **6,** 1–238 (2012) (cit. on p. 44).

85. Kaufmann, B., Leone, N., Perri, S. & Schaub, T., Grounding and Solving in Answer Set Programming, *AI Magazine* **37,** 25–32, ISSN: 2371-9621, https://ojs.aaai.org/index.php/aimagazine/article/view/2672 (Oct. 2016) (cit. on p. 44).

86. Syrjänen, T., Omega-Restricted Logic Programs, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **2173 LNAI,** 267–280, https://link.springer.com/chapter/10.1007/3-540-45402-0_20 (2001) (cit. on p. 44).

87. Wittocx, J., Mariën, M. & Denecker, M., Grounding FO and FO(ID) with Bounds, *Journal of Artificial Intelligence Research* **38,** 223–269, ISSN: 1076-9757, https://www.jair.org/index.php/jair/article/view/10653 (May 2010) (cit. on p. 44).

88. Simons, P., Niemelä, I. & Soininen, T., Extending and implementing the stable model semantics, *Artificial Intelligence* **138,** 181–234, ISSN: 0004-3702 (June 2002) (cit. on p. 44).

89. Lin, F. & Zhao, Y., ASSAT: computing answer sets of a logic program by SAT solvers, *Artificial Intelligence* **157,** 115–137, ISSN: 0004-3702 (Aug. 2004) (cit. on p. 44).

90. Gebser, M., Kaminski, R., König, A. & Schaub, T., Advances in gringo Series 3, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **6645 LNAI,** 345–351, https://link.springer.com/chapter/10.1007/978-3-642-20895-9_39 (2011) (cit. on p. 44).

91. Gebser, M., Kaufmann, B., Neumann, A. & Schaub, T., clasp: A Conflict-Driven Answer Set Solver, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **4483 LNAI,** 260–265, https://link.springer.com/chapter/10.1007/978-3-540-72200-7_23 (2007) (cit. on p. 44).

92. Lefebvre, M., Gaignard, A., Folschette, M., Bourdon, J. & Guziolowski, C., Large-scale regulatory and signaling network assembly through linked open data, *Database* **2021** (2021) (cit. on pp. 48 sq., 65, 77).

93. Rodchenkov, I., Babur, O., Luna, A., Aksoy, B. A., Wong, J. V., Fong, D., Franz, M., Siper, M. C., Cheung, M., Wrana, M., Mistry, H., Mosier, L., Dlin, J., Wen, Q., O'Callaghan, C., Li, W., Elder, G., Smith, P. T., Dallago, C., Cerami, E., Gross, B., Dogrusoz, U., Demir, E., Bader, G. D. & Sander, C., Pathway Commons 2019 Update: integration, analysis and exploration of pathway data, *Nucleic Acids Research* **48,** D489–D497, ISSN: 0305-1048, eprint: https://academic.oup.com/nar/article-pdf/48/D1/D489/31697765/gkz946_supplemental_file.pdf (Oct. 2019) (cit. on pp. 48, 61, 65).

94. Wrzodek, C., Büchel, F., Ruff, M., Dräger, A. & Zell, A., Precise generation of systems biology models from KEGG pathways, *BMC Systems Biology* **7,** 15, ISSN: 1752-0509, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3623889/ (2024) (Feb. 2013) (cit. on p. 48).

95. Fabregat, A., Jupe, S., Matthews, L., Sidiropoulos, K., Gillespie, M., Garapati, P., Haw, R., Jassal, B., Korninger, F., May, B., Milacic, M., Roca, C. D., Rothfels, K., Sevilla, C., Shamovsky, V., Shorser, S., Varusai, T., Viteri, G., Weiser, J., Wu, G., Stein, L., Hermjakob, H. & D'Eustachio, P., The Reactome Pathway Knowledgebase, eng, *Nucleic Acids Research* **46,** D649–D655, ISSN: 1362-4962 (Jan. 2018) (cit. on p. 48).

96. Schaefer, C. F., Anthony, K., Krupa, S., Buchoff, J., Day, M., Hannay, T. & Buetow, K. H., PID: the Pathway Interaction Database, eng, *Nucleic Acids Research* **37,** D674–679, ISSN: 1362-4962 (Jan. 2009) (cit. on p. 48).

97. Guziolowski, C., Videla, S., Eduati, F., Thiele, S., Cokelaer, T., Siegel, A. & Saez-Rodriguez, J., Exhaustively characterizing feasible logic models of a signaling network using Answer Set Programming, *Bioinformatics* **29,** 2320–2326, `https://academic.oup.com/bioinformatics/article/29/18/2320/240224` (2013) (cit. on pp. 50, 52 sq., 61).

98. Videla, S., Guziolowski, C., Eduati, F., Thiele, S., Gebser, M., Nicolas, J., Saez-Rodriguez, J., Schaub, T. & Siegel, A., Learning Boolean logic models of signaling networks with ASP, *Theoretical Computer Science* **599,** 79–101 (2015) (cit. on pp. 50, 52).

99. Videla, S., Saez-Rodriguez, J., Guziolowski, C., Siegel, A. & Wren, J., caspo: a toolbox for automated reasoning on the response of logical signaling networks families, *Bioinformatics* **33,** 947–950, ISSN: 33/6/947/2585024 (2017) (cit. on pp. 50, 52 sq., 55, 61, 72, 78, 80).

100. Alexopoulos, L. G., Saez-Rodriguez, J., Cosgrove, B. D., Lauffenburger, D. A. & Sorger, P. K., Networks Inferred from Biochemical Data Reveal Profound Differences in Toll-like Receptor and Inflammatory Signaling between Normal and Transformed Hepatocytes*, *Molecular & Cellular Proteomics* **9,** 1849–1865, ISSN: 1535-9476, `https://www.sciencedirect.com/science/article/pii/S1535947620309014` (2024) (Sept. 2010) (cit. on p. 51).

101. Videla, S., Konokotina, I., Alexopoulos, L. G., Saez-Rodriguez, J., Schaub, T., Siegel, A. & Guziolowski, C., Designing experiments to discriminate families of logic models, *Frontiers in Bioengineering and Biotechnology* **3,** 131–131 (2015) (cit. on p. 53).

102. Huynh-Thu, V. A., Irrthum, A., Wehenkel, L. & Geurts, P., Inferring Regulatory Networks from Expression Data Using Tree-Based Methods, en, *PLOS ONE* **5,** e12776, ISSN: 1932-6203, `https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0012776` (2024) (Sept. 2010) (cit. on p. 125).

103. Horvath, S., Zhang, B., Carlson, M., Lu, K. V., Zhu, S., Felciano, R. M., Laurance, M. F., Zhao, W., Qi, S., Chen, Z., Lee, Y., Scheck, A. C., Liau, L. M., Wu, H., Geschwind, D. H., Febbo, P. G., Kornblum, H. I., Cloughesy, T. F., Nelson, S. F. & Mischel, P. S., Analysis of oncogenic signaling networks in glioblastoma identifies ASPM as a molecular target, *Proceedings of the National Academy of Sciences of the United States of America* **103,** 17402–17407, ISSN: 0027-8424, `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1635024/` (2024) (Nov. 2006) (cit. on p. 129).

104. Gargalovic, P. S., Imura, M., Zhang, B., Gharavi, N. M., Clark, M. J., Pagnon, J., Yang, W.-P., He, A., Truong, A., Patel, S., Nelson, S. F., Horvath, S., Berliner, J. A., Kirchgessner, T. G. & Lusis, A. J., Identification of inflammatory gene modules based on variations of human endothelial cell responses to oxidized lipids, *Proceedings of the National Academy of Sciences of the United States of America* **103,** 12741–12746, ISSN: 0027-8424, `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1568918/` (2024) (Aug. 2006) (cit. on p. 129).

COLLEGE ED MATHEMATIQUES
DOCTORAL STIC
PAYS DE LA LOIRE

Nantes Université

Titre : Programmes logiques pour déduire des modèles informatiques du développement embryonnaire humain

Mot clés : Réseaux Booléens – Answer-Set Programming – Biologie des Systèmes – Développement Préimplantatoire Humain – Single-cell transcriptomics

Résumé : L'avènement des nouvelles technologies de séquençage sur cellule unique permet aujourd'hui une analyse fine et approfondie des mécanismes de régulation génique impliqués dans le développement embryonnaire humain. Cet avancement ouvre de nouvelles perspectives pour étudier ce processus complexe et encore mal compris. Cette thèse s'intéresse à la modélisation informatique du développement embryonnaire, dans le but de mieux comprendre les mécanismes de régulation génique impliqués. Pour ce faire, nous présentons SCIBORG, une méthode qui infère des réseaux Booléens modélisant des stades de développement d'intérêt. En combinant des interactions géniques issues de connaissances préalables avec l'expression génique observée dans les cellules d'embryons, des modèles Booléens sont appris pour indiquer les mécanismes impliqués dans les stades de développement étudiés. Nous avons appliqué notre méthode à deux stades de développement du trophectoderme et découvert des mécanismes de régulation distincts dans nos modèles. Dans l'ensemble, cette thèse propose une méthode de modélisation des mécanismes de régulation impliqués dans un processus de différenciation cellulaire, offrant ainsi un outil précieux pour l'étude du développement embryonnaire humain.

Title: Logic programs to infer computational models of human embryonic development

Keywords: Boolean Networks – Answer-Set Programming – Systems Biology – Human Preimplantation Development – Single-cell transcriptomics

Abstract: The advent of new single-cell sequencing technologies now allows for a detailed and in-depth analysis of the gene regulatory mechanisms involved in human embryonic development. This advancement opens new perspectives in the complex and limited study of embryonic development. This thesis focuses on the computational modeling of development to better understand the gene regulatory mechanisms involved. To this end, we present SCIBORG, a method that infers Boolean networks modeling the stages of development of interest. By combining gene interactions from prior knowledge with observed gene expression in embryonic cells, Boolean models are learned, indicating the mechanisms involved in the studied developmental stages. We applied our method to two developmental stages of the trophectoderm and discovered distinct regulatory mechanisms in our models. Overall, this thesis proposes a method for modeling the regulatory mechanisms involved in a cell differentiation process, providing a valuable tool for studying human embryonic development.