DOCTORAL DISSERTATION OF THE UNIVERSITY OF BURGUNDY FRANCHE-COMTE PREPARED AT UNIVERSITY OF BURGUNDY

Doctoral School n°37

Sciences Physiques pour l'Ingénieur et Microtechniques (SPIM)

PhD in Computer Science

By

Mrs. Carine KHALIL

Study of combinatorial statistics and their impact in evolutionary optimization

A dissertation presented and defended at Dijon, on 12/11/2021

Composition of the Jury :

Mr Kondo ADJALLAH	Professor, University of Lorraine	Reviewer
Mr Hamamache KHEDDOUCI	Professor, University of Lyon 1	Reviewer
Mrs Christelle BLOCH	Associate Professor, University of Franche-Comté	Examiner
Mr Vlady RAVELOMANANA	Professor, University of Paris	Examiner
Mr Phan THUAN DO	Associate Professor, Hanoï University of Science and Technology	Examiner
Mr Vincent VAJNOVSZKI	Professor, University of Burgundy	Co-supervisor
Mr Jean-Luc BARIL	Professor, University of Burgundy	Co-supervisor
Mr Wahabou ABDOU	Associate Professor, University of Burgundy	Co-supervisor

Acknowledgments

This thesis could not have been completed without the continuous support and encouragement of all those who accompanied me during these three years of research.

I begin by expressing my great gratitude to my three thesis co-supervisors **Vincent Vajnovzki**, **Jean-Luc Baril**, and **Wahabou Abdou**. It is thanks to your experience and your encouragement and your time that I was able to carry out my research work in the right direction despite my moments of doubt. You provided me with all the necessary and sufficient conditions, scientific support, very enriching exchanges, and permanent availability which have enabled me to broaden my vision and improve my research. It was a pleasure working with you. This thesis was funded by the ministry and supervised by Ecole Doctorale SPIM for which I am also grateful.

also, I express my gratitude to Mr. **Kondo Adjallah**, Professor at the University of Lorraine, and Mr. **Hamamache Kheddouci**, Professor at the University of Lyon 1, for accepting of being reviewers for this thesis to evaluate this work and for devoting part of their time to it. I would also like to thank Ms. **Christelle Bloch**, Mr. **Vlady Ravelomanana**, and Mr. **Phan Thuan Do** for agreeing to be examiners for this thesis.

Also, I would like to thank all the members of the Laboratoire d'Informatique de Bourgogne for always being available to exchange some discussions, and for the fraternal atmosphere that reigns there. In addition, I thank my friends and colleagues in Dijon with whom I shared good times through outings and evenings organized on all occasions.

My last thanks and not the least go to my family, especially my parents, **Zahi** and **Carmen**, and my brothers **Mohamad** and **Youssef**, for their unconditional support and love, their curiosity for my research, even if it is not always understandable, and for their boundless pride in me. Finally, **Ali**, all of this thesis work relies heavily on your understanding, support, and encouragement from the start and until the last moment!

Carine

Abstract

This thesis studies combinatorial objects, with both an algorithmic and a combinatorial point of view. In the combinatorial part, we take care first, the enumeration of Catalan words avoiding pairs of patterns of length three, presenting the proofs of each case with various enumeration methods. Catalan words are particular growth-restricted words counted by the eponymous integer sequence. More precisely, we systematically explore the structural properties of the sets of words under consideration and give enumerating results by constructive bijections or bivariate generating functions with respect to the length and descent number. Then, we study a sorting machine using two stacks in series where the first one avoids a pair of patterns of length three. The process consists of a right-greedy algorithm since we apply at each step the first possible transformation. In this dissertation, we primarily focus on the machines which sort a set of permutations enumerated by Catalan and Schröder numbers. For each class of such permutations, we give a characterization in terms of avoiding patterns which allow us to provide the exact enumeration.

In the second part, being more experimental, we studied the optimization of permutation problems with genetic algorithms. Different kinds of encoding of the solutions have been considered to study the transmission of genetic properties in permutation problems. In particular, we used the Lehmer code, inversion table encoding, transposition array encoding, and inverse transposition array encoding during the process. Solution encoding describes the way decision variables are represented. Since indirect encodings are not sensitive to duplicates, they lead to a loss of genetic properties during crossbreeding. This contribution proposes a study of the impact of this loss both in the space of decision variables and in that of objective functions considering the four indirect encodings. In addition, after analyzing that the use of the Lehmer code and inverse transposition array preserve schemata after the recombination process, we proposed an adaptive crossover operator in order to be able to keep the prefix/suffix in order to see its influence on the transmission of genetic properties.

Resumé

Ces travaux de thèse portent sur l'étude des objets combinatoires, à la fois d'un point de vue algorithmique et combinatoire. Nous nous occupons d'abord, dans la partie combinatoire, de l'énumération des mots de Catalan en évitant les paires de motifs de longueur trois, en présentant les preuves de chaque cas avec différentes méthodes d'énumération. Les mots de Catalan sont des mots particuliers à croissance restreinte comptés par la séquence entière éponyme. Plus précisément, nous explorons systématiquement les propriétés structurelles des ensembles de mots considérés et donnons des résultats d'énumération par bijections constructives ou par des fonctions génératrices bivariées par rapport à la longueur et au nombre de descentes. Ensuite, nous étudions une machine de tri à deux piles en séries où la première pile évite une paire de motifs de longueur 3. Ce tri résulte d'un algorithme glouton puisqu'on réalise à chaque étape la première opération possible. Dans cette thèse, nous nous intéressons en particulier aux machines de tri pour lesquelles les permutations triables sont comptées par les nombres de Catalan et les nombres de Schröder. Pour chaque classe de ces permutations, nous donnons une caractérisation en termes de motifs exclus ce qui nous permet de fournir des résultats exacts d'énumération.

Dans la seconde partie, plus expérimentale, nous étudions l'optimisation des problèmes de permutations avec des algorithmes génétiques. Différents types de codage de solutions sont mis en œuvre pour étudier la transmission de certaines propriétés génétiques dans les problèmes de permutations. En particulier, nous étudions le code de Lehmer, les tables d'inversion, les tableaux de transposition et les tableaux de transposition inverses. Le codage des solutions décrit la manière dont les variables de décision sont représentées. Les codages indirects ne sont pas sensibles aux doublons, cependant ils conduisent à une perte des propriétés génétiques lors des croisements. Cette contribution propose une étude de l'impact de cette perte à la fois dans l'espace des variables de décision et dans celui des valeurs des fonctions objectifs considérant les quatre codages indirects. De plus, après avoir analysé l'utilisation des codages indirects code de Lehmer et tableaux de transposition inverses, une préservation de schémas après le processus de croisement a retenu notre attention. Une adaptation sur l'opérateur de croisement a été réalisée afin de pouvoir conserver le préfixe / suffixe et d'appréhender son influence sur la transmission des propriétés génétiques.

Contents

List of Figures

List of Symbols

$\mathcal{A}_n$	the set of ascent sequences of length n
A_n	$\lvert\mathcal{A}_n\rvert$
$\mathcal{A}(x)$	the g.f. for the ascent sequences
$asc(w)$	the number of ascents in the word w
$b.g.f.$	bivariate generating function
BP	barred pattern
$\mathcal{C}(x)$	the g.f. for the Catalan numbers
c_n	the n-th Catalan number
$\mathcal{D}_n$	the set of all Dyck paths with $2n$ steps
$des(w)$	the number of descents in the word w
F_n	the n-th Fibonacci number with $F_0 = 1$, $F_1 = 1$
$g.f.$	generating function
$i(\pi)$	the inverse of a permutation π
$inv(\pi)$	the number of inversions in a permutation π
$\mathcal{IS}_n$	the set of inversion sequences of length n
$[k]$	the set $\{1, 2, \ldots, k\}$
$[k]^n$	the set of all words of length n over $[k]$
$\mathcal{M}(x)$	the g.f. for the Motzkin numbers
M_n	the n-th Motzkin number
$\mathbb{N}$	the set of natural numbers $\{0, 1, 2, 3, \ldots\}$
n-permutation	permutation of length n
$\mathbb{R}$	the set of real numbers
$\mathcal{S}(x)$	the g.f. for the Schröder numbers
S_n	the n-th (large) Schröder number
$\mathfrak{S}_n$	the set of all permutations of length n
$\mathfrak{S}_n(P)$	the set of all n-permutations avoiding each pattern in P
$s_n(P)$	$\lvert\mathfrak{S}_n(P)\rvert$
$Sort_n(\sigma)$	the set of n-permutations sortable with the σ-machine
VP	vincular pattern
$\mathcal{W}_n(k)$	the set of all words of length n over $[k]$, $[k]^n$
$W_n(k, P)$	the set of all words in $[k]^n$ that avoid each pattern in P
$\mathrm{W}_n(k, P)$	$\lvert W_n(k, P)\rvert$
$W^t(n)$	the set of t-stack-sortable permutations in $\mathfrak{S}_n$
$\mathcal{W}^t(n, k)$	set of of t-stack-sortable permutations in $\mathfrak{S}_n$ with number of descents equal to k
σ-machine	the right greedy algorithm performed on two stacks in series, such that the first stack is σ-avoiding and the second stack is 21-avoiding.

Résumé étendu

Introduction

Les travaux présentés dans cette thèse se situent à la confluence de deux domaines de l'informatique théorique, qui sont d'une part la combinatoire énumérative et d'autre part l'optimisation évolutionnaire.

La combinatoire est une branche des mathématiques discrètes qui étudie des collections d'objets discrets. Certains de ses buts sont de caractériser les objets étudiés, d'établir leurs propriétés, de compter, pour tout n, le nombre d'objets de taille n, ou de les générer de manière exhaustive ou aléatoire. Plus précisément, la combinatoire énumérative est généralement définie comme la discipline qui traite le problème du comptage des éléments dans un ensemble fini. Les objets étudiés dans cette thèse sont les permutations et les mots, considérés principalement du point de vue de l'inclusion/évitement des motifs dans les permutations et les mots. Dans la première partie de cette thèse, on s'intéresse sur deux classes combinatoires : les mots de Catalans évitant des motifs et des permutations triables par une (σ, τ)-machine, en collectant des résultats énumératifs, algébriques et algorithmiques sur les deux classes.

Du point de vue de l'optimisation évolutionnaire, dans la deuxième partie de cette thèse, les permutations sont utilisées dans de nombreux problèmes d'optimisation combinatoire (COP) pour représenter des solutions candidates. De tels problèmes sont communément appelés problèmes de permutations. Les problèmes de permutations peuvent être trouvés dans divers domaines d'applications tels que l'ordonnancement, le routage, etc. Nos travaux utilisent des algorithmes évolutionnaires (EA), plus précisement, des algorithmes génétiques (AG) qui sont des métaheuristiques inspirées de la théorie de l'évolution. Une population de solutions initiales appelées individus évolue au fil des générations grâce à l'utilisation de certains opérateurs génétiques tel que la sélection, le croisement, et la mutation.

Les objets formels étudiés en combinatoire et en algorithmique sont toujours des modèles d'objets réels, même si la modélisation apparaît plus ou moins immédiatement selon les problèmes étudiés. Les motivations pour l'étude de ces objets proviennent de l'informatique elle-même (modélisation de structures de données, analyse de réseaux, ...). Dans cette thèse, nous verrons que la motivation la plus immédiate provient de l'intérêt pour les mécanismes qui sous-tendent le processus d'optimisation évolutionnaire.

Dans l'étude d'une famille d'objets discrets, la combinatoire et les algorithmes génétiques offrent deux points de vue sur ces objets complémentaires : les résultats combinatoires obtenus peuvent avoir un intérêt en optimisation, et certains opérateurs en algorithm génétique peuvent être intéressant d'un point de vue combinatoire. En effet, les propriétés combinatoires des objets, et

en particulier les propriétés mettant en évidence une structure dans ces objets, peuvent être utilisées pour concevoir des algorithmes qui, tirant parti de ces propriétés, sont plus complexes qu'un algorithme sans cette connaissance a priori combinatoire.

Combinatoire énumérative

La première partie de la thèse est centrée sur le domaine de la combinatoire énumérative. La combinatoire, également appelée mathématiques combinatoires, est la partie des mathématiques discrètes qui étudie des ensembles finis d'objets appelés objets combinatoires. Le problème de base de la combinatoire énumérative est de compter des ensembles avec certaines contraintes. Cependant, ces derniers temps, ce domaine a trouvé une variété d'utilisations, à la fois en sciences pures et appliquées, et par conséquent, il est devenu une partie du courant dominant des mathématiques modernes. De grands noms tels que Donald Knuth, Philippe Flajolet, Richard Stanley, André Joyal ont contribué à la création de fondations de combinatoire.

Mots de Catalans évitant des motifs

Ainsi après que Baril et al. [23] aient étudié la distribution des descentes sur des ensembles de mots de Catalan évitant un motif de longueur au plus 3, nous avons étudié les connexions entre séquences de l'Encyclopédie en ligne des séquences entières [213] et ces mots évitant deux motifs de longueur 3. Les mots de Catalan sont des mots particuliers sur l'ensemble des entiers non négatifs à croissance restreinte et ils représentent une classe combinatoire comptée par les nombres Catalan, voir [199, 6.19.u, p. 222].

Le tableau 2 resume tous les résultats obtenus dans cette contribution. Il y avait beaucoup de cas triviaux, tels que les *motifs superflus*, *suites ultimement constantes* et la *suite* n. De nombreux cas ont été dénombrés par comptage par récurrence, principalement via la séquence de comptage $2(n-1)$, des séquences impliquant 2^n, des séquences impliquant des coefficients binomiaux et des séquences impliquant des nombres de type Fibonacci. Il est important de noter que pour certains cas, les suites n'ont pas été étudiées dans la littérature, elles ont donc été comptées en donnant des fonctions génératrices bivariées $C_\pi(x, y)$ où le coefficient de $x^n y^k$ est le nombre de mots de Catalan. de longueur n ayant k descentes et évitant π. Des propositions et des corollaires ont également été proposés pour correspondre à certains cas, en utilisant des séquences d'ascension. Nous avons spécialement utilisé pour les énumérations le corollaire 1 qui dit, pour $n \geq 4$ et un motif π de longueur trois, $\mathcal{A}_n(\pi) = \mathcal{C}_n(\pi)$ si et seulement si $\pi \in \{001, 010, 012, 102\}$.

Tri de permutations avec des piles sous contraintes

L'étude du tri de permutations (classement des permutations par ordre croissant) a commencé avec Knuth [124], qui analyse un certain "algorithme de tri par piles", puis il a introduit la composition en série de piles et a donné de nombreux exemples de liens entre le comportement pire / meilleur / moyen des algorithmes de tri et les structures combinatoires cachées derrière les permutations [125].

Puisque le problème (classique) de caractérisation et d'énumération des permutations qui peuvent

$\sigma\backslash\tau$	000	001	010	011	012	021	100	101	102	110	120	201	210
000	-	P. 14	P. 14	$u.c.$	$u.c.$	*C. 2*	s	P. 7	*C. 9*	*C. 8*	*C. 10*	*C. 9*	*C. 14*
001	-	-	P. 5	P. 5	P. 5	P. 13	P. 15	s	s	P. 13	P. 13	s	P. 12
010	-	-	-	P. 5	P. 5	s	s	s	s	s	s	s	s
011	-	-	-	-	P. 5	s	P. 6	s	s	s	P. 6	s	s
012	-	-	-	-	-	s	P. 13	P. 13	s	P. 13	s	s	s
021	-	-	-	-	-	-	P. 9	P. 9	P. 11	*C. 4*	P. 10	s	s
100	-	-	-	-	-	-	-	P. 17	*C. 13*	*C. 7*	C. 3	P. 16	C. 15
101	-	-	-	-	-	-	-	-	s	P. 9	P. 9	s	P. 8
102	-	-	-	-	-	-	-	-	-	P. 11	P. 8	*C. 6*	*C. 12*
110	-	-	-	-	-	-	-	-	-	-	C. 3	*C. 5*	s
120	-	-	-	-	-	-	-	-	-	-	-	s	s
201	-	-	-	-	-	-	-	-	-	-	-	-	*C. 11*
210	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 2: Les paires $\{\sigma, \tau\}$ où τ est superflu pour σ sont marquées par s et celles produisant des suites d'énumération constantes par $u.c.$. Les références sont aux propositions ou aux corollaires où les suites d'énumération ou des fonctions génératrices sont données. Les paires référencées par la même proposition ou corollaire forment une classe d'équivalence de Wilf et les résultats d'énumération qui ne sont pas encore enregistrés dans [2] sont en italique. Les paires mises en évidence sont déjà énumérées dans [30] dans le contexte des séquences d'ascension, voir la section 3.1.

être triées à de deux piles connectées en série est encore largement ouvert, Cerbai et al. ont présenté dans [60] un problème connexe, dans lequel ils ont imposé des restrictions à la fois sur la procédure et sur les piles. Plus précisément, ils considéraient un algorithme glouton où ils exécutaient les opérations les plus à droite (ici "le plus à droite" fait référence à la représentation habituelle des problèmes de tri de pile).

Dans cette contribution, nous traiterons de machines de tri similaires constituées de deux piles connectés en série. En rappelant les propriétés clés de l'algorithme Stack-sort, nous considérerons les machines obéissant à certaines contraintes où la première pile évite (σ, τ), une paire de motifs de longueur trois et la deuxième pile évite le motif 21. Après [60], nous l'appelons (σ, τ)-machine. Plus précisément, nous nous limitons aux paires de motifs pour lesquels les permutations triables sont comptées soit par les nombres catalans, soit par deux de leurs proches parents : la transformée binomiale des nombres de Catalan et les nombres de Schröder.

Pour la paire (132.231) nous avons montré que les permutations triables sont celles évitant 1324 et 2314, un ensemble dont l'énumération est donnée par les grands nombres de Schröder. Sous certaines conditions sur les motifs évités, la sortie de la première pile est bijectivement liée à son entrée (voir [33, 59]) : il s'ensuit que pour trois paires de motifs, à savoir $(123, 213)$, $(132, 312)$ et $(231, 321)$, les permutations triables sont comptées par les nombres de Catalan. Ce résultat a été prouvé indépendamment dans [21, 33]. Pour la paire $(123, 132)$, nous avons prouvé que les permutations triables sont celles évitant les motifs 2314, 3214, 4213 et le motif généralisé $[24\bar{1}3$ Aussi, nous avons prouvé que les permutations triables sont énumérées par les nombres de Catalan en montrant que la distribution du premier élément est donnée par le triangle de Catalan bien connu. Nous avons montré que pour la paire $(123, 312)$ la séquence de comptage correspondante est la transformée binomiale des nombres de Catalan.

Optimization evolutionnaire

La première partie de cette thèse a porté sur la combinatoire énumérative. Elle a pris en compte les propriétés combinatoires de certains objets combinatoires, en particulier les propriétés de permutations. Dans la deuxième partie, en tant que travail complémentaire, nous étudions l'application des propriétés des permutations dans l'optimisation évolutionnaire. En effet, la contribution proposée concerne les problèmes de permutations et les algorithmes génétiques (plus particulièrement le codage et les opérateurs génétiques). L'optimisation des problèmes de permutations est largement étudiées dans la littérature en raison de leur complexité et de la diversité de leurs domaines d'application. Résoudre un tel problème consiste à trouver une permutation qui minimise / maximise certains critères.

Transmission des propriétés génétiques dans les problèmes de permutations

Lorsqu'on traite des problèmes de permutations, il faut s'assurer qu'il n'y a pas de doublons dans la permutation. Cependant, au fil des générations, les opérateurs génétiques tels que le croisement et la mutation peuvent dupliquer les allèles (valeurs). Il existe principalement deux approches pour éviter ces répétitions. La première consiste à utiliser des opérateurs de croisement et de mutation qui réparent les individus contenant des doublons (par exemple OX, UX ou PMX). Une autre méthode consiste à utiliser un code (dit indirect) qui tolère les doublons. On définit une bijection entre ce codage indirect et la permutation. Le code de Lehmer et la table d'inversion sont des exemples de codage indirect. De plus, deux encodages non classiques sont présentés dans cette contribution. Inspiré des travaux de J-L. Baril, nous nous intéressons au tableau de transposition présenté dans [19]. Une autre version de ce codage est proposée: le tableau de transposition inverse.

Cependant, il convient de noter qu'une partie des propriétés génétiques des parents peut être perdue lors de la génération des enfants si le codage indirect est utilisé. Cette contribution étudie la transmission de propriétés génétiques en se concentrant à la fois sur les variables de décision et la fonction objectif. Nous considérons le problème de voyageur de commerce (TSP) comme exemple de problème de permutations. Pour étudier la transmission des caractéristiques des parents aux enfants, nous utilisons des métriques qui se concentrent sur les similitudes entre les solutions (Hamming Distance, Position Based Indicator, Edge Based Indicator). Ensuite, pour analyser la distribution des solution de chaque codage par rapport à la fonction objectif, une méthode est utilisée, inspirée de l'étude faite par Portman et Vignier dans [183]. Ils classent les solutions, pour chaque génération, en cinq groupes en fonction des valeurs de fitness. Trois problèmes TSP sont étudiés dans cette thèse: `eil51`, `att48` et `burma14`.

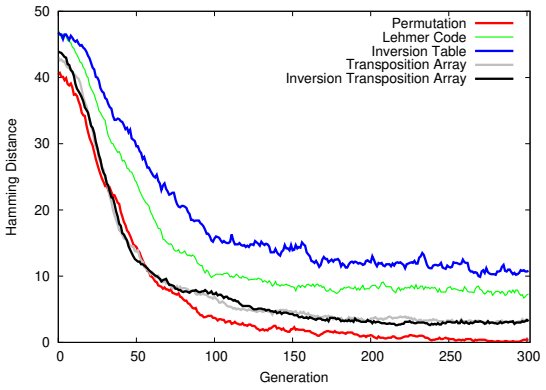


Figure 1: Hamming Distance pour `eil51`

Les résultats montrent que le codage indirect permet de préserver la diversité au sein des populations. Plus préciément, fig. 7.3 montre la performance de la distance de Hamming entre les parents et les enfants sur des générations pour les problèmes `eil51`. On observe que cette métrique diminue progressivement au fil des générations car l'algorithme génétique aura tendance à exploiter le voisinage des meilleures solutions. Cependant, en raison de l'apparition de nouveaux allèles, les codages indirects montrent de bonnes valeurs HD et parviennent tout de même à diminuer et à rester supérieurs au codage direct qui tend vers zéro. Cette diversité est accompagnée par une conservation de qualité au niveau de la fonction objectif. Les solutions appartenant au groupe 1 dans le cas du codage indirect sont majoritaires et la population reste diversifiée.

Étude de conservation de schéma dans les problèmes de permutations

Dans cette contribution, nous étudions la transmission des propriétés génétiques lors de l'utilisation de codages indirects. Nous nous concentrons sur le code de Lehmer et le codage du tableau de transposition inverse car ils préservent le préfixe / suffixe après le processus de recombinaison. Cela nous rappelle la théorie des schémas de Holland [116], le fondement des explications de la puissance des algorithmes génétiques (AG) et l'une des premières tentatives pour comprendre comment ces algorithmes fonctionnent dans un sens formel. On suppose que, grâce au processus de reproduction entre individus, les schémas les plus adaptés sont plus susceptibles d'être dans la génération suivante. Une certaine adaptation est proposée pour être appliquée au cours du processus de croisement qui aide à conserver le schéma de bons parents.

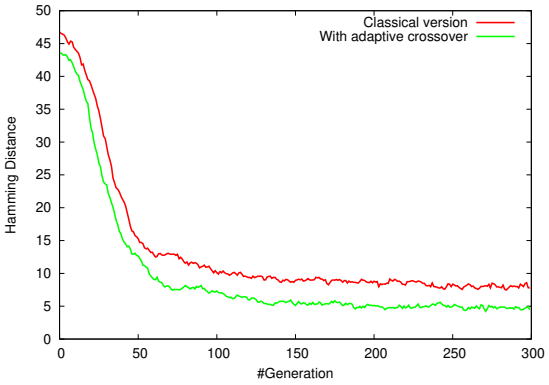


Figure 2: Hamming Distance utilisant Lehmer code

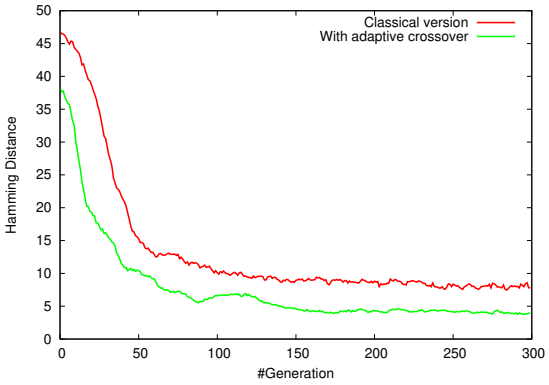


Figure 3: Hamming Distance utilisant Inverse transposition array

Le recours à l'adaptation lors du croisement a permis de conserver les allèles des parents. La conservation est forte pour les meilleurs individus. Parce que ces meilleurs individus sont ceux qui survivent et se reproduisent, cette préservation sera donc présente au fil des générations. Cela se reflète bien dans les valeurs des métriques utilisées, notamment par exemple dans les valeures de Hamming distance dans les fig. 2 et 3.

Chapter 1

Introduction

1.1 Context

The work presented in this thesis lies at the confluence of two fields of theoretical computer science, which are on the one hand enumerative combinatorics and on the other hand evolutionary computation.

Combinatorics is a branch of discrete mathematics that studies collections of discrete objects, i.e. for which we have a notion of a size such that the number of objects of size n is finite for each n. Some of the goals of combinatorics are to characterize the studied objects, to shed light on their properties, to count, for all n, the number of objects of size n, or to generate them exhaustively or randomly. More specifically, enumerative combinatorics is usually defined as the discipline that deals with the problem of counting the elements in a finite set. This is of course a somewhat coarse definition, as the notion of counting, is not entirely transparent itself. In fact, a finite set $\mathcal{A}$ is assigned as the subset of all the elements in a broader finite set $\mathcal{B}$ satisfying a given characteristic property $\mathcal{P}$. Therefore, given an explicit list of the elements of $\mathcal{B}$, to count the elements in $\mathcal{A}$ one can simply scan the list of the elements of $\mathcal{B}$, check which elements satisfy the property $\mathcal{P}$ to produce an explicit list of the elements of $\mathcal{A}$, attach the label 1 to the first element of $\mathcal{A}$ you find, the label 2 to the second one, and so on. The last label one attaches is the number of elements in $\mathcal{A}$. However, it is easy to see that it quickly becomes insoluble in terms of computation, depending on the parameters characterizing the finite set, so we must have a good answer for this type of problem. A more precise discussion could get philosophical, depending on the criteria one can choose to consider a counting algorithm as computationally efficient. Usually, one is given a sequence of finite sets $\{\mathcal{A}_n\}_{n\in\mathbb{N}}$ and is asked to count the number $f(n)$ of elements in $\mathcal{A}_n$ for every $n \in \mathbb{N}$. The objects of study in this thesis are permutations and words, considered mainly from the point of view of pattern involvement/avoidance in permutations and words.

The study of patterns in discrete structures is currently one of the most active trends in combinatorics research. The notion of a pattern in a permutation was historically born from the problem of sorting permutations with certain devices. However, the richness of this notion has become particularly evident from its numerous appearances in several very different disciplines, such as algebra, geometry, analysis, theoretical computer science, biology, and many others. Similar notions of pattern have been considered on discrete structures other than permutations, such as integer sequences and words, lattice paths, graphs, mappings, and set partitions. The first part of this thesis aims to shed light on two combinatorial classes: Catalan words avoiding patterns and permutations

sortable by a (σ, τ)-machine, by collecting enumerative, algebraic, and algorithmic results on both classes. Most of the enumerating results are obtained by generating functions.

Permutations are also used in many combinatorial optimization problems (COPs) to represent candidate solutions. Such problems are commonly known as permutation-based problems. They are studied in the second part of this thesis which focuses on evolutionary computation. Permutation-based problems can be found in various application areas such as scheduling and sequencing, design of VLSI circuits, campus design, routing, etc. They consist either of the optimization of the assignment cost between two sets of objects of the same cardinality, or the optimization of a schedule of jobs or any other planning application represented as a sequence of tasks, and so on. Such problems are generally NP-hard and some of them are among the most challenging optimization problems in terms of resolution time. Their complexity is due partially to the exponential size of their search space, which makes large benchmarks very difficult to be solved. Our results take place in the context of evolutionary algorithms (EAs), in particular Genetic algorithms (GAs) which are very popular population-based metaheuristics inspired by the theory of evolution. A population of initial configurations called individuals is evolved through generations by the use of some genetic operators by analogy to their counterparts in biological systems: selection, crossover, mutation, etc. This method is a very promising research area for solving large permutation-based problems.

1.2 Motivation

The formal objects studied in combinatorics and in algorithmics are always models of real objects, even if the modelization appears more or less immediately according to the problems studied. The motivations for the study of these objects come from computer science itself (modeling of data structures, network analysis, ...). In this thesis, we will see that the most immediate motivation comes from optimization problems involved in the study of evolutionary computation.

In the study of a family of discrete objects, combinatorics and genetic algorithms offer two points of view on these complementary objects: the combinatorial results obtained can have optimization repercussions, and some genetic algorithms have implications from a combinatorial point of view. Indeed, the combinatorial properties of objects, and in particular the properties highlighting a structure in these objects, can be used to design algorithms. Taking advantage of these properties, design algorithms are more complex than one without this a priori combinatorial knowledge.

1.3 Detailed outline and contributions

This thesis is organized into two parts. **Part I** is about enumerative combinatorics and is composed of three chapters. **Part II** contains three chapters about evolutionary optimization and permutation-based optimization problems. The manuscript is ended with some concluding remarks and future research directions.

Part I: Enumerative combinatorics

Chapter 2 is an introduction to enumerative combinatorics. It essentially constitutes a state of the art on combinatorial objects and enumeration methods. This review is completed by presenting interesting related works. This chapter deals almost exclusively with the combinatorics of

different classes. We formally define the permutations as well as some interesting lattice paths, sequences, and words. We also give some very classical properties of these objects. Then the notion of pattern, classical and non-classical, is defined in the context of statistics on combinatorial classes. This chapter ends with listing and explaining different enumeration methods in particularly generating function and recurrence relation.

Chapter 3 presents our first contribution, namely the enumeration of Catalan words avoiding pairs of patterns of length three, presenting the proofs of each case with different enumeration methods. Catalan words are particular growth-restricted words counted by the eponymous integer sequence. In this contribution, we consider Catalan words avoiding a pair of patterns of length 3, pursuing the recent initiating work of J.-L. Baril et al. [23], where (among other things) the enumeration of Catalan words avoiding patterns of length 3 is completed. More precisely, we explore systematically the structural properties of the sets of words under consideration and give enumerating results by constructive bijections or bivariate generating functions with respect to the length and descent number. Some of the obtained enumerating sequences are known, and thus the corresponding results establish new combinatorial interpretations for them. The results presented in this chapter are published in Discrete Mathematics & Theoretical Computer Science journal, and in Permutation Patterns 2019 conference [22].

Chapter 4 is about stack sorting. We present in this chapter our second contribution about sorting with restricted stacks avoiding some pairs of patterns. Pattern avoiding machines were introduced recently by Claesson et al. [60] as a particular case of the two-stacks in series sorting device. They consist of two restricted stacks in series, ruled by a right-greedy procedure and the stacks avoid some specified patterns. Some of the obtained results have been further generalized to Cayley permutations by Cerbai [59], specialized to particular patterns by Defant and Zheng [77], or considered in the context of functions over the symmetric group by Berlow [33]. In this work, we study pattern avoiding machines where the first stack avoids a pair of patterns of length 3 and investigate those pairs for which sortable permutations are counted by the (binomial transform of the) Catalan numbers and the Schröder numbers.

Part II: Evolutionary Optimization

Chapter 5 is an introduction to permutation-based problems and evolutionary algorithms. A non-exhaustive list of permutation-based problems is given, as well as the explanation of the genetic algorithm and its operators. This review is completed by an analysis of the interpretation of the permutation encoding. Some mathematical properties of permutations, tools and indicators are also presented.

Chapter 6 presents the third contribution of this thesis. It studies the transmission of genetic properties in permutation problems using Lehmer code, inversion table encoding, transposition array encoding and inversion transposition array encoding. Solution encoding describes the way decision variables are represented. In the case of permutation problems, the classical encoding should ensure that there are no duplicates. During crossover operations, repairs may be carried out to correct or avoid repetitions. The use of indirect encoding aims to define bijections between the classical permutation and a different representation of the decision variables. These encodings are

not sensitive to duplicates. However, they lead to a loss of genetic properties during crossbreeding. This contribution proposes a study of the impact of this loss both in the space of decision variables and in the fitness values domain, considering four indirect encodings: Lehmer code, Inversion table, Transposition array and Inversion transposition array.

In **Chapter 7**, we study the transmission of genetic properties when using indirect encodings. We focus on the Lehmer code and the Inverse transposition array encoding as they preserve the prefix / suffix after the recombination process. This reminds us of the schema theory of Holland [116], the foundation for explanations of the power of genetic algorithms (GAs) and one of the earliest attempts to understand how these algorithms work in a formal sense. It supposes that, through the process of recombination, the fittest schemata are more likely to be in the next generation. Some adaptation is proposed to be applied during the crossover process which helps to retain the schema of good parents to the offspring. To do this, we will study the conservation properties of the suffix or prefix for the Lehmer code and the Inverse transposition table.

Part I

Enumerative combinatorics

Chapter 2

Notations, definitions and useful results in Enumerative combinatorics

This part of the thesis is considered in the field of enumerative combinatorics. Combinatorics, also called combinatorial mathematics, is the part of discrete mathematics that studies finite sets of objects called combinatorial objects. The basic problem of enumerative combinatorics is counting sets with certain constraints. It is a discipline that has developed through various branches of mathematics such as number theory, group theory, probability theory, integer series, complex analysis, algebraic geometry. However, in recent times this field has found a variety of uses, both in pure and applied science, and as a result, it has become part of the mainstream of modern mathematics. Big names such as Donald Knuth, Philippe Flajolet, Richard Stanley, André Joyal have contributed to the creation of foundations of combinatorics. In this chapter, we will present some introductory notions in enumerative combinatorics, as well as some of the methods and tools used in this thesis, but we must start by defining the basic objects on which this study relates in particular combinatorial objects and their statistics.

Let $n \neq 0$ be an element of $\mathbb{N}$, we denote by $[n]$ the set $\{1, 2 \dots\}$. For two integers a and b, $a \leq b$, the interval $[a, b]$ consists of the set $\{a, a+1, \dots, b\}$. When $b < a$, the interval $[a, b]$ is empty.

2.1 Combinatorial classes

Definition 1. A combinatorial class is a collection $\mathcal{O}$ of objects of a similar kind (e.g. sequences, words, trees, graphs, permutations, paths, ...), endowed with a suitable notion of size $p : \mathcal{O} \to \mathbb{N}$ such that there are only finitely many objects of each size, i.e. $|p^{-1}(n)|$ is finite for all $n \in \mathbb{N}$. For convenience, we consider that there is a unique object of size 0, and this object will be called $\emptyset$.

An important part of combinatorial problems is to enumerate the number of objects of a given size in a combinatorial class. Below, we define all combinatorial classes used in this thesis, and we provide some classical enumeration methods (see the books [114, 122, 132, 167, 199] for more details).

2.1.1 Sequences and words

Integer sequences

Definition 2. Formally, a sequence with values in a set B is a function $a : I \to B$, where $I \subseteq \mathbb{N}$. We denote it by $a = (a_n)_{n\in I}$ where $a_n = a(n)$, and whenever I consists of $i_1, i_2, \ldots, i_n, \ldots$ (sorted in increasing order), we write $a = a_{i_1}, a_{i_2}, \ldots, a_{i_n}, \ldots$.

The set I is called the set of indices, and generally $I = \mathbb{N}$, or $I = \{0, 1, 2, \ldots, n\}$ for some $n \in \mathbb{N}$. A sequence is said to be finite when the set of indices is finite. In this thesis, infinite sequences are used for the enumerations of classes of combinatorial objects, that is $a_n = |p^{-1}(n)|$ where p is the size of objects.

Maybe the prototypical integer sequence is that of celebrated Fibonacci numbers, $(F_n)_{n\geq 0}$, sequence A000045 in [2], with its first values $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots$.

The Fibonacci sequence is defined recursively by $F_0 = 0$, $F_1 = 1$ and for $n \geq 2$

$$F_n = F_{n-1} + F_{n-2}.$$

A closed-form formula does exist and is given for $n \geq 0$ by

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right).$$

The generating function for the sequence $(F_n)_{n\geq 0}$ is $\mathcal{F}(x) = \sum_{n\geq 0} F_n x^n$ given by

$$\mathcal{F}(x) = \frac{x}{1 - x - x^2}.$$

Other ubiquitous integer sequences in combinatorics are Catalan numbers and their close relatives Motzkin and Schröder numbers. The sequence of Catalan numbers defined by

$$c_n = \frac{1}{n+1}\binom{2n}{n},$$

is referred as A000108 in [2], with its first values $1, 1, 2, 5, 14, 42, 132, 429, \ldots$ It counts a large variety of combinatorial objects, for instance, ordered rooted trees with n nodes, Dyck paths and words, and Catalan words defined below. In [200], Richard Stanley gives more than 200 mostly combinatorial interpretations of the sequence $(c_n)_{n\geq 0}$.

The generating function for the Catalan numbers, defined by $\mathcal{C}(x) = \sum_{n\geq 0} c_n x^n$, satisfies

$$\mathcal{C}(x) = 1 + x\mathcal{C}^2(x)$$

with the solution

$$\mathcal{C}(x) = \frac{1 - \sqrt{1-4x}}{2x}.$$

Motzkin numbers $(M_n)_{n\geq 0}$, sequence A001006 in [2] satisfy, for $n \geq 2$, the recurrence relation

$$M_n = \frac{2n+1}{n+2}M_{n-1} + \frac{3n-3}{n+2}M_{n-2},$$

with its first values $1, 1, 2, 4, 9, 21, 51, 127, 323, \ldots$ They are also expressed in the term of binomial coefficient and Catalan numbers as

$$M_n = \sum_{k=0}^{[\frac{n}{2}]} \binom{n}{2k} C_k.$$

The generating function $\mathcal{M}(x) = \sum_{n\geq 0} M_n x^n$ for the Motzkin numbers satisfies

$$\mathcal{M}(x) = 1 + x\mathcal{M}(x) + x^2\mathcal{M}^2(x)$$

with the solution

$$\mathcal{M}(x) = \frac{1 - x - \sqrt{1-2x-3x^2}}{2x^2}.$$

Finally, the large Schröder numbers S_n is the sequence A006318 in [2]. It satisfies the recurrence relation

$$S_n = 3S_{n-1} + \sum_{k=1}^{n-2} S_k S_{n-k-1}$$

for $n \geq 2$ with $S_0 = 1$, $S_1 = 2$. The corresponding generating function $\mathcal{S}(x) = \sum_{n=0} S_n x^n$ is

$$\mathcal{S}(x) = \frac{1 - x - \sqrt{x^2-6x+1}}{2x}.$$

Words

In this thesis, we also consider finite sequences as objects in a combinatorial class, which can also be viewed as a word on an alphabet (by deleting commas in the notation $a_0, a_1, a_2, \ldots$). Below, we briefly introduce the basic terminology on words. For a more detailed introduction in the field, see Lothaire [142], and Berstel and Perrin [34] seminal books.

Definition 3. A word of length n, $w = w_1 w_2 \ldots w_n$, is a sequence whose symbols (or letters) come from a set A called an *alphabet*. Alphabets in this thesis are finite, and the most typical alphabets are of the form $\{0, 1, \ldots, k-1\}$ or $[k] = \{1, \ldots, k\}$, depending on the content.

For $n \geq 0$ and $k \geq 0$, let $\mathcal{W}_n(k)$ be the set of words of length n over $[k]$ and $\mathcal{W}(k) = \bigcup_{n\geq 0} \mathcal{W}_n(k)$. There are k^n different words of length n over a k-letter alphabet: $|\mathcal{W}_n(k)| = k^n$. For example, 2411121 and 25554 are words over the alphabet $[5] = \{1, 2, 3, 4, 5\}$ while *abbacca* is a word over the alphabet $\{a, b, c\}$. Also, the complete set of words in $[2]^3$ is $W_3(2) = \{111, 112, 121, 211, 122, 212, 221, 222\}$. The first systematic study of words seems to have appeared in three papers of Axel Thue (1863-1922) in 1906, 1912 and 1914 [210–212], and in a paper by MacMahon in 1913 [144]. Thue's work is on infinite words (sequence of symbols) and he investigated words from a number-theoretic viewpoint. By contrast, MacMahon approach words, which he called *assemblage of objects*, in the context of partitions and permutations. His viewpoint and questions are the first instance of enumerating statistics on words.

Catalan word

Catalan words are particular growth-restricted words and they represent a combinatorial class counted by the Catalan numbers, see for instance exercise [199, 6.19.u, p. 222]. See also in [149], Catalan words are considered in the context of the exhaustive generation of Gray codes for growth-restricted words.

Definition 4. A length n Catalan word is a word $w = w_1 w_2 \dots w_n$ over the set of non-negative integers with $w_1 = 0$, and $0 \leq w_i \leq w_{i-1} + 1$ for $i = 2, 3, \dots n$.

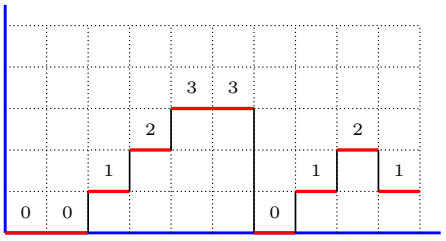


Figure 2.1: A path corresponding to the Catalan word 0012330121

From a Catalan word $w_1 w_2 \dots w_n$, we construct a path in the first quadrant of the plane, starting at the origin and ending at $(n, 0)$, consisting of horizontal steps $H = (1, 0)$ and vertical steps $(0, \pm 1)$ such that the i-th horizontal step lies on the line $y = w_i$. Fig. 2.1 represents an example of a path corresponding to a Catalan word. We denote by $\mathcal{C}_n$ the set of length n Catalan words, and $\mathcal{C} = \bigcup_{n \geq 0} \mathcal{C}_n$. For instance, $\mathcal{C}_2 = \{00, 01\}$ and $\mathcal{C}_3 = \{000, 001, 010, 011, 012\}$. It is well known that the cardinality of $\mathcal{C}_n$ is given by the n-th Catalan number

$$c_n = \frac{1}{n+1} \binom{2n}{n}$$

which is the general term of the sequence A000108 in the On-line Encyclopedia of Integer Sequences [2].

Ascent sequence

Ascent sequences are sequences of non-negative integers with restrictions on the size of each letter, depending on the number of ascents preceding it in the sequence. Given a word $w = w_1 w_2 \dots w_n$ over the alphabet $\mathbb{N}$, $asc(w_1 \dots w_n)$ will be the number of ascents in the sequence w, that is, the number of places $j \geq 1$ such that $w_j < w_{j+1}$.

Definition 5. An ascent sequence is a sequence $w = w_1 w_2 \dots w_n$ of nonnegative integers satisfying $w_1 = 0$, and, for all i with $1 < i \leq n$, $w_i \leq asc(w_1 \dots w_{i-1}) + 1$.

We write $\mathcal{A}_n$ for the set of ascent sequences of length n and we let $A_n = |\mathcal{A}_n|$. The number of ascent sequences of length n is given by the Fishburn numbers, A022493 in [2], with the first values $1, 1, 2, 5, 15, 53, 217, 1014, 5335, 31240, 201608, \dots$. For example, 01234, 0120102, and 01013 are all ascent sequences, while 01024 is not since $asc(0102) = 2$ and we do not have $4 \leq asc(0102) + 1$. Ascent sequences have been an increasingly frequent topic of study since Bousquet-Melou, Claesson, Dukes, and Kitaev related them to $(2+2)$-free posets and enumerated the total number of ascent sequences [43], who also managed to count these, which was quite a feat. They proved that the generating function of ascent sequences is

$$\mathcal{A}(x) = \sum_{n\geq 0} \prod_{i=1}^{n} (1-(1-x)^i).$$

Ascent sequences have since been studied in a series of papers by various authors, connecting them to many other combinatorial structures. These connections, and generalizations of them, have exposed what seem to be deep structural correspondences in these apparently disparate combinatorial objects. A good source of references and further information is in [66,87,88]. Obviously, a Catalan word is an ascent sequence, and we reveal some connections between the two classes of combinatorial objects.

Dyck word

Dyck words, or well-parenthesed words, are a central object in combinatorics and formal language theory. They are enumerated by Catalan numbers as many other objects. For instance, Chapter 6 of [199] points out the role of Dyck words in enumerative combinatorics. They have been studied extensively, and there are well-known bijections between Dyck words and other combinatorial objects such as binary trees, parallelogram polyominoes, 2-lines standard tableaux and so on.
Let $B = \{0, 1\}$ be a binary alphabet and $w = w_1 w_2 \dots w_n \in B^n$. Let $h : B \to \{-1, 1\}$ be a valuation function with $h(0) = 1$, $h(1) = -1$, and

$$h(w_1 w_2 \dots w_n) = \sum_{i=1}^{n} h(w_i).$$

Definition 6. A semi-length n word $w = w_1 w_2 \dots w_{2n} \in B^{2n}$ is called a Dyck word if $h(w_1 w_2 \dots w_i) \geq 0$, for $1 \leq i \leq 2n-1$ and $h(w_1 w_2 \dots w_{2n}) = 0$, see [86].

We shall denote by $\mathcal{D}_n$ the set of Dyck words of length $2n$, and by $\mathcal{D} = \bigcup_{n\geq 0} \mathcal{D}_n$ the class of all Dyck words. The number of Dyck words of length $2n$, $D_n = |\mathcal{D}_n|$, as previously mentioned, is the n-th Catalan number

$$c_n = \frac{1}{n+1}\binom{2n}{n}.$$

For example, there are 14 Dyck words for $n = 4$: 00001111, 00010111, 00011011, 00011101, 00100111, 00101011, 00101101, 00110011, 00110101, 01000111, 01001011, 01001101,01010011, 01010101.

2.1.2 Permutations

The now-famous free encyclopedia Wikipedia opens its article devoted to permutations with the following definition:

> *In mathematics, a permutation of a set is, loosely speaking, an arrangement of its members into a sequence or linear order, or if the set is already ordered, a rearrangement of its elements. The word "permutation" also refers to the act or process of changing the linear order of an ordered set.*

This definition is immediately followed by an anchoring of the permutations in the context in which they are studied:

Permutations differ from combinations, which are selections of some members of a set regardless of order.

In this thesis, we are interested in permutations of an interval of integers, usually starting with 1.

Definition 7. A permutation of length $n \in \mathbb{N}$, also referred to as an n-permutation, is a bijection from the set $[n]$ to itself.

We write permutations as words $w = w_1 w_2 \dots w_n$, whose letters are distinct and usually consist of the integers $1, 2, 3, \dots, n$, where w_i is the image of i in w. We denote by $\mathfrak{S}_n$ the set of permutations of length n and $\mathfrak{S} = \bigcup_{n\in\mathbb{N}} \mathfrak{S}_n$. It is well known that there are $n!$ distinct n-permutations, $|\mathfrak{S}_n| = n!$. When $n = 0$, then $\mathfrak{S}_n$ is the empty set. We denote by $id_n = 1 \dots n$ the identity permutation of $[n]$.
For example, 413265 is a 6-permutation, and $\mathfrak{S}_1 = \{1\}$, $\mathfrak{S}_2 = \{12, 21\}$, $\mathfrak{S}_3 = \{123, 132, 213, 231, 312, 321\}$. The graphical representation of $w \in \mathfrak{S}_n$ is the set of points in a Euclidean plane, provided with an orthonormal system, whose coordinates are the pairs (i, w_i) with $i \in [n]$. See Fig. 2.2 for an example.

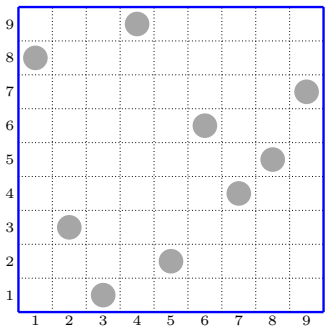


Figure 2.2: Graphical representation of the permutation $w = 831926457$

2.1.3 Lattice paths

As another example of combinatorial class, we exhibit the hugely studied case of lattice paths. A lattice path is intuitively defined as its name suggests: a path (or walk) in a lattice in a Euclidean plane ε provided with an origin reference $\mathbb{O}$. We denote by the letters **u**, **d**, and **f** (like Up, Down, and Flat) the vectors of respective coordinates $(1, 1)$, $(1, -1)$ and $(1, 0)$ called respectively step up, step down and plateau. Paths are better for visual intuition and words are better for writing proofs, but of course, they represent in different ways the same combinatorial object.
An insight into the study of generic lattice paths and their close link with Probability and Statistics is reported in [131] and in [157], as well as a discussion about the basic methods for counting lattice paths. In literature (see for instance [14]), precise computable estimates are given for the number of lattice paths under various constraints: with ending point lying on the x-axis (bridges) or, constrained to remain in the positive quarter plane (meanders) or both conditions at the same time (excursions). Below is a brief presentation on the families of lattice paths frequently involved in our study.

Dyck path

Definition 8. A Dyck path of length $2n$, $n \in \mathbb{N}$ is a lattice path in $\mathbb{Z}^2$ between $(0, 0)$ and $(2n, 0)$ consisting of up-steps $(1, 1)$ and down-steps $(1, -1)$ which never goes below the x-axis.

It is convenient to encode each up-step by the letter **u** or number 1 and each down step by **d** or 0, obtaining an encoding of the Dyck path as a Dyck word. The set of Dyck paths of length $2n$, like the set of Dyck words defined previously, is denoted by $\mathcal{D}_n$. As an example, the Dyck path corresponding to the Dyck word **uduuuududddd** is drawn in Fig. 2.3.

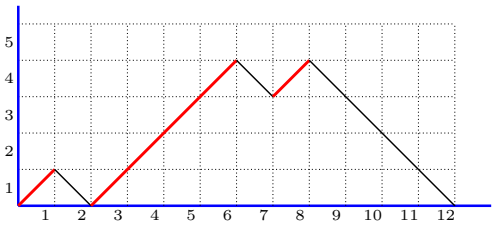


Figure 2.3: A path corresponding to the Dyck word 101111010000 or **uduuuududddd**

Among the structures counted by Catalan numbers, an important role is played by Dyck paths, for which a vast literature is devoted. Different aspects of these combinatorial objects are studied, such as their random generation, the bijections with other combinatorial structures, the enumeration according to a great variety of parameters, giving rise to nice statistics involving well-known numbers like Narayana number (see [202, 203], for instance). We refer to [79] for some enumerative results and a vast bibliography concerning the enumeration of Dyck paths.

Motzkin path

Motzkin paths are simple combinatorial objects that appear in many contexts. They are similar to Dyck paths but allow also horizontal (flat) steps of unit length.

Definition 9. A Motzkin path of length $n \in \mathbb{N}$ is a lattice path starting at point $(0, 0)$, ending at $(n, 0)$, and never going below the x-axis, consisting of up steps $\mathbf{u} = (1, 1)$, down steps $\mathbf{d} = (1, -1)$ and horizontal steps $\mathbf{h} = (1, 0)$.

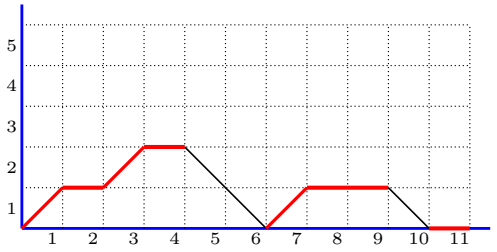


Figure 2.4: A path corresponding to the Motzkin word **uhudduhhdu**

Let $\mathcal{M}_n$ be the set of Motzkin paths of length n and we set $\mathcal{M} = \cup_{n\geq 0}\mathcal{M}_n$. See Fig. 2.4 for an illustration of a Motzkin path of length 11. There are many studies on Motzkin paths [46,79,84,163]. For example, Baril and Petrossian [27] conducted the study of the equivalence relation in the language of Motzkin's words on the alphabet $\{\mathbf{u}, \mathbf{d}, \mathbf{h}\}$. These lattice paths are counted by the Motzkin numbers, sequence A001006 in [2], which the first terms are $1, 1, 2, 4, 9, 21, 51, 127, 323, 835, \ldots$.

Schröder path

Definition 10. A Schröder path of length $n \in \mathbb{N}$ is a lattice path from $(0, 0)$, to $(2n, 0)$, and never going below the x-axis, consisting of up steps $\mathbf{u} = (1, 1)$, down steps $\mathbf{d} = (1, -1)$ and flat

steps $\mathbf{h} = (2, 0)$.

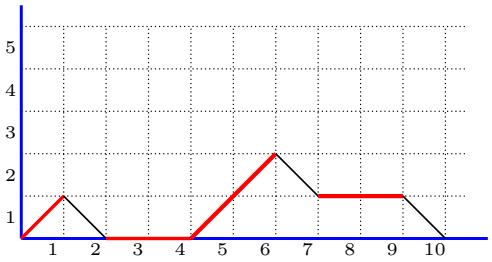


Figure 2.5: A path corresponding to a Schröder word

Step $(2, 0)$ is usually called the double horizontal step and it is denoted by $\mathbf{h}_2$ or simply by $\mathbf{h}$ when no risk of confusion with Motzkin paths arises. As mentioned before, it is well known that Schröder paths are enumerated by the (large) Schröder numbers, which form sequence A006318 in [2]. For other combinatorial objects counted by the Schröder numbers, see [17, 126, 187].

2.2 Statistics on combinatorial classes

The subject of permutation statistics, it is frequently claimed, dates back at least to Euler [94]. However, it was not until MacMahon's extensive study [144] at the turn of the century that this became an established discipline of mathematics, and it was to take a long time before it developed into the vast field that it is today. In the past three decades or so much progress has been made, both in discovering and analyzing new statistics and in extending these, together with the classical permutation statistics, to arbitrary words, see [93, 97, , 198].

Definition 11. Given $\mathcal{O}$ a class of combinatorial objects, a statistic on $\mathcal{O}$ is a map $f : \mathcal{O} \to \mathbb{N}$. The distribution of f is the sequence $(a_i)_{i\in\mathbb{N}}$ where a_i is the cardinality of $f^{-1}(i)$.

Note that the length function (also called size) of a set of combinatorial objects is an obvious statistic.

MacMahon considered four different statistics for a permutation p: The number of descents $des\ p$, the number of exceedances $exc\ p$, the number of inversions $inv\ p$, and the major index $maj\ p$.

These are defined as follows:

$$\begin{aligned} des\ p &= \text{number of descents in } p, \\ exc\ p &= \text{number of excedances in } p, \\ inv\ p &= \text{number of } (i, j) \text{ such that } i < j \text{ and } p_i > p_j, \\ maj\ p &= \text{sum of the descents in } p. \end{aligned}$$

In fact, MacMahon studied these statistics in greater generality, namely over the rearrangement class of words w. All of the above-mentioned statistics generalize to words, and in each case, except for that of exc, the generalization is trivial.

Also, statistics often involve the notion of patterns contained in a combinatorial object. That is why we give below definition and several examples of patterns for words and permutations.

2.2.1 Patterns

There are countless notions of patterns in different combinatorial objects. Considering words and permutations, one meets many notions of patterns in these objects in the literature. Roughly speaking, patterns in permutations are permutations with additional restrictions, and patterns in words are certain restricted words, possibly permutations.

Definition 12. Given a word $\pi = \pi_1\pi_2\dots\pi_k$, we say a word $w = w_1w_2\dots w_n$ contains the pattern π, $k \leq n$, if there is a sub-word of w, $w_{i_1}\dots w_{i_k}$, order-isomorphic with $\pi_1\pi_2\dots\pi_k$. If w does not contain π, we say w avoids π.

See for instance Kitaev's seminal book [122] on this topic. For example, $w = 3422155$ contains the pattern 211 because the letters 322 have the same relative order in w as 211. On the other hand, 3422155 avoids pattern 4321. Let the patterns π_1 and π_2, if π_2 contains π_1, then the π_2 restriction is superfluous since every π_1-avoiding permutation is also π_2-avoiding.

For $n, k \geq 0$ and a set P of patterns, we denote by $W_n(k, P)$ the set of words of length n over the alphabet $[k]$ which avoid every pattern in P.

Classical Patterns

Classical patterns, introduced for permutations by Knuth [124] in 1968 but studied intensively for the first time by Simion and Schmidt [193] in 1985; this notion was extended to words by Burstein [52] in 1998, and generalized for words (by allowing repetitions in patterns) by Burstein and Mansour [55] in 2002; two books [39, 40] by Bona appeared in 2004 and 2006 which discuss classical patterns for permutations, and the book [114] by Heubach and Mansour appeared in 2010 and discusses these patterns for words.

- Patterns in permutations: Let the set $\mathfrak{S}_n(\pi)$ consist of all length n permutations that avoid π. If P is a set of permutations/patterns, we define $\mathfrak{S}_n(P) = \bigcap_{\pi\in P} \mathfrak{S}_n(\pi)$. So $\mathfrak{S}_n(P)$ consists of all length n permutations that avoid every member of P. The problem of finding $s_n(P)$, the cardinality of $\mathfrak{S}_n(P)$, for various patterns has received much attention. The first two calculations were $s_n(123)$ and $s_n(132)$, by MacMahon [152] and Knuth [126] respectively. Both cardinalities turn out to be the n-th Catalan number A000108. Later, Simion and Schmidt [193] found $s_n(P)$ for all $P \subseteq \mathfrak{S}_3$.

  This was followed by several articles that found $s_n(\{\pi_1, \pi_2\})$ for various pairs of permutations: Billey, Jockusch, and Stanley [36] and West [224] solved the problem for $\pi_1 \in \mathfrak{S}_3$, $\pi_2 \in \mathfrak{S}_4$, and Kremer and Shiu [133] considered several cases with $\pi_1, \pi_2 \in \mathfrak{S}_4$.

- Patterns in ascent sequences: We write $\mathcal{A}_n(P)$ for the set of ascent sequences of length n avoiding all patterns in P. Also, we let $A_n(P) = |\mathcal{A}_n(P)|$. Pattern avoidance in ascent sequence was first studied by Duncan and Steingrímsson [91]. They studied the enumeration of sequences avoiding patterns of length 3 and 4 with the entries in the Online Encyclopedia of Integer Sequences [2]. Mansour and Shattuck [147] later computed the number of sequences avoiding 1012 and 0123 and showed that certain statistics on 0012-avoiding ascent sequences are equidistributed with other statistics on the set of 132-avoiding permutations.

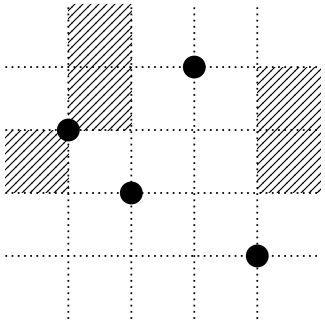
Non-classical patterns

Barred patterns: First introduced for permutations by West [221], a barred pattern is specified by a permutation with some barred entries. If $\bar{\pi}$ is a barred pattern, let π be the permutation obtained by removing all bars of $\bar{\pi}$, and let π' be the permutation which is order isomorphic to the non-barred entries in $\bar{\pi}$. For example, let $\bar{\pi} = 4\bar{3}2\bar{1}5$, so $\pi = 43215$ and $\pi' = 312$. An occurrence of barred pattern $\bar{\pi}$ in a permutation w is then an occurrence of π' in w that is not part of an occurrence of π in w. Conversely, for w to avoid $\bar{\pi}$, each occurrence in w of π' must appear in an occurrence of π. Moreover, barred patterns in words do not seem to appear in the literature, even though this notion is well defined.

Vincular patterns: Vincular patterns were introduced for permutations by Babson and Steingrímsson [13] in 2000; this notion was extended to words by Burstein and Mansour [54] in 2003; the book [114] by Heubach and Mansour discusses vincular patterns for words. A vincular or generalised pattern specifies the conditions for adjacency. Two different notations are used. Traditionally, a vincular pattern is written as a permutation with dashes inserted between terms that need not be adjacent and no dashes between terms that that need to be adjacent. Alternatively, and perhaps preferably, terms which are to be adjacent are underlined. For example, 314265 contains two occurrences of $2\underline{31}4$ (or $2-31-4$) and a single occurrence of $2\underline{314}$ ($2-314$) but avoids $\underline{23}\,\underline{14}$ ($23-14$). A vincular pattern in which all the terms must appear contiguously is called a consecutive pattern.

Bivincular patterns: In a bivincular pattern, conditions of vincular patterns are also placed on which entries must take adjacent (consecutive) values. It was introduced for permutations by Bousquet-Mélou Claesson, Dukes and Kitaev [43] in 2010. This notion can not be extended directly to words.

Mesh patterns: Mesh patterns were introduced by Brändén and Claesson in [49] to provide explicit expansions for certain permutation statistics as (possibly infinite) linear combinations of (classical) permutation patterns. This notion was further studied by Kitaev and Remmel in some series of papers refining conditions on permutations and patterns, see for instance [123]. Classical, vincular and bivincular patterns are all examples of the more general family of mesh patterns. Formally, a mesh pattern p of length k is a pair (π, R) with $\pi \in \mathfrak{S}_k$ and $R \subseteq [0,k] \times [0,k]$, a set of pairs of integers. An example is $p = (3241, \{(0,2),(1,3),(1,4),(4,2),(4,3)\})$. To depict this mesh pattern, we plot the points (i, w_i) in a Cartesian coordinate system, and for each $(i,j) \in R$ we shade the unit square with the bottom left corner (i,j):



An occurrence of a mesh pattern (π, R) in a permutation w consists of an occurrence of the classical pattern π in w such that no elements of w occur in the shaded regions of the figure. A vincular pattern is therefore a mesh pattern in which complete columns are shaded.

2.3 Enumeration methods

Combinatorics is the art of counting. The general object of enumerative combinatorics is to calculate the cardinal for particular sets. Most of the time, the set depends on one or more parameters, and it is a question of expressing this cardinal as a function of these parameters.

In combinatorics, there are powerful tools that help us to solve many enumerative problems, namely recurrence relation and generating function. Recurrence relation arises naturally in definitions of sequences or when we try to break the problem into cases, some of which are similar to the original problem. Generating functions are closely related to sequences, and can be used to solve recurrence relations and other kinds of problems such as enumeration problems and proving identities involving sequences. In this thesis, we will have some results and applications using recurrence relations and generating functions among other enumeration tools.

2.3.1 Bijection

A useful and combinatorially meaningful method to compute the number of elements in a set is to provide a bijection with a set already counted. Bijective Combinatorics is a field that consists in studying the enumerative properties of sets of combinatorial objects by exhibiting bijections (ideally explicit), which preserve some properties, between such sets and already known objects. This then makes it possible to apply all the tools of analytical combinatorics to these new objects, in order to obtain an explicit enumeration, asymptotic properties, or even to generate them exhaustively. We present below examples of a bijection between sets of combinatorial objects

Dyck words vs. Catalan words

Many different classes of combinatorial objects are enumerated by the well-known Catalan numbers. This is the case, among others, of ballot sequences, planar trees, Young tableaux, stack sortable permutations, etc. A list of over 60 types of such combinatorial classes of independent interest was compiled by Stanley [199]. There is also in the literature a certain number of explicit bijections between these Catalan classes.

There is a direct bijection

$\delta \mapsto w$ between maybe the most celebrated combinatorial class enumerated by Catalan numbers, Dyck paths of semi length n, and Catalan words $\mathcal{C}_n$. Indeed, in a length $2n$ Dyck path collecting for the up steps the ordinates of their starting points we obtain a length n Catalan word, and this construction is a bijection. See Fig. 2.6 where this bijection is depicted for an example.

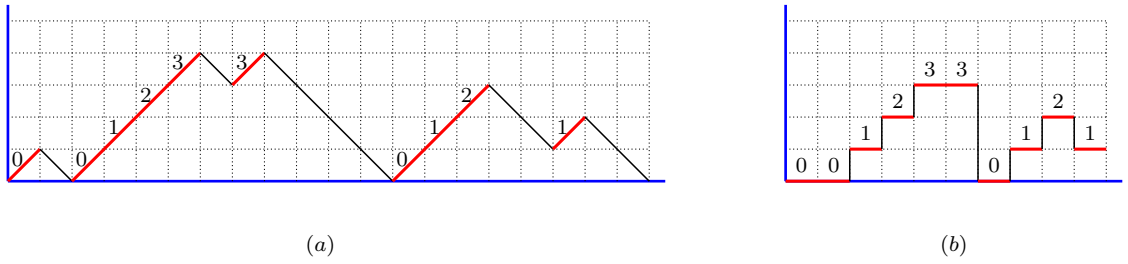


Figure 2.6: (a) The Dyck path **uduuuududddduuuddudd** where each up step is labeled by the ordinate of its starting point; and (b) its corresponding Catalan word 0012330121

Note that the above bijection gives a one-to-one correspondence between occurrences of the consecutive pattern **ddu** in Dyck words and descents in Catalan words.

132-Avoiding permutations vs. Dyck paths

In [130], Krattenthaler exhibited a Dyck path correspondence ϕ for 132-avoiding permutations. Let ϕ the map from the set of 132-avoiding permutations to Dyck paths as follows. Let $w = w_1 w_2 \dots w_n$ be a 132-avoiding permutation. We read the permutation w from left to right and successively generate a Dyck path. When w_j is read, then in the path we adjoin as many up-steps as necessary, followed by a down-step from height $h_j + 1$ to height h_j (measured from the x-axis), where h_j is the number of elements in $w_{j+1} \dots w_n$ which are larger than w_j.
For example, let $w = 74352681$. The first element to be read is 7. There is one element in 4352681 that is larger than 7; therefore the path starts with two up-steps followed by a down-step, thus reaching height 1 (see Fig. 2.7).

The next 4 is read. Three elements in 352681 are larger than 4; therefore the path continues with three up-steps followed by a down-step, thus reaching height 3, etc. The complete Dyck path $\phi(74352681)$ is shown in Fig. 2.7.

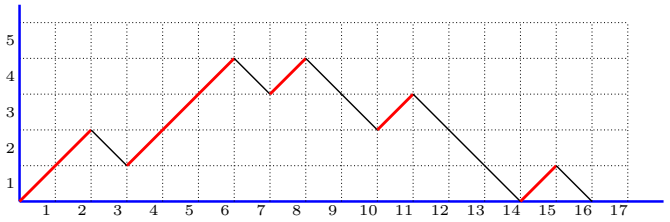


Figure 2.7: The Dyck path corresponding to 74352681

2.3.2 Recurrence relation

A *recurrence relation* is a formula that recursively defines a sequence $(a_n)_{n=0}^{\infty}$ provided some values of initial terms $a_m, a_{m+1}, \dots a_{m+k}$ are given, where m, k are non-negative integers. In other words, subsequent terms $a_{m+k+1}, a_{m+k+2}, \dots$ are defined as a function of the preceding terms. For example, let F_n denote the Fibonacci numbers having recurrence $F_n = F_{n-1} + F_{n-2}$, $n \geq 2$, with the initial values $F_0 = 0$ and $F_1 = 1$. A word $w_1, w_2, \dots, w_n$ is an inversion sequence if $0 \leq w_i < i$ for all $i \in [n]$, see [70]. Let $\mathcal{IS}_n(\pi)$ the set of inversion sequences of length n which avoid the pattern π. Mansour and Shattuck [147] counted the set $\mathcal{IS}_n(012)$ and proved that it is counted by the odd indices Fibonacci numbers F_{2n-1}.

2.3.3 Generating function

Let $\mathcal{W}$ be a set of words and for each $n \geq 0$, let $\mathcal{W}_n$ be the set of words in $\mathcal{W}$ of length n. The *generating function* for the set $\mathcal{W}$ is the formal power series

$$\mathcal{W}(x) = \sum_{n \geq 0} |\mathcal{W}_n| x^n = \sum_{\boldsymbol{w} \in \mathcal{W}} x^{|\boldsymbol{w}|}.$$

In particular, when π is a pattern, $\mathcal{W}(\pi)$ is the set of words in $\mathcal{W}$ avoiding π, and $\mathcal{W}_n(\pi)$ the set of words of length n in $\mathcal{W}(\pi)$, then the generating function for $\mathcal{W}(\pi)$ is the formal power series

$$\mathcal{W}_\pi(x) = \sum_{n \geq 0} |\mathcal{W}_n(\pi)| x^n = \sum_{\boldsymbol{w} \in \mathcal{W}(\pi)} x^{|\boldsymbol{w}|}.$$

More general, let $f_{k,P}$ be the generating function for words over the alphabet $[k] = \{1, \ldots, k\}$ avoiding the pattern set P. That is, $f_{k,P}(x) = \sum_{n \geq 0} |\mathcal{W}_n(k, P)| x^n$ where $\mathcal{W}_n(k, P)$ is precisely the set of length n words over $[k]$ and avoiding P. In order to better illustrate the generating fuctions as an enumerating tool, we consider the following 'degenerate' cases:

- If $1 \in P$, then $f_{k,P}(x) = 1$ (only the empty word avoids the pattern 1)
- If $P = \varnothing$, then $f_{k,P}(x) = \frac{1}{1-kx}$ (the generating function for all words over $[k]$).

Sometimes, we deal with multivariate generating functions. For instance, let $\mathcal{W}(x, y)$ be the bivariate generating function where the coefficient of $x^n y^k$ is the number of pattern-avoiding words of length n and a fixed value k for a given statistic, say s. Then

$$\mathcal{W}(x, y) = \sum_{\boldsymbol{w} \in \mathcal{W}} x^{|\boldsymbol{w}|} y^{s\,\boldsymbol{w}}.$$

From $\mathcal{W}(x, y)$, simple substitutions will allow computing either the associated univariate generating functions or other similar variations. For example, Asinowski, Bacher, Banderier, and Gittenberger [11] were interested in enumerating the lattice paths that avoid a fixed pattern π, and in counting the occurrences of a given pattern. So they used the bivariate generating function $\mathcal{W}(t, u)$, where the variable t encodes the length of the paths, and the variable u the final altitude. And also, they used the trivariate generating function with the variables length, final altitude, and the number of occurrences of the pattern π. To show the effectiveness of generating functions as a counting tool, we give below two examples for counting Dyck and Motzkin paths according to their length. Many studies on Motzkin and Dyck paths appear in the literature. Generally, they consist in enumerating these paths according to several parameters, such as length, height, number of occurrences of a pattern, number of returns to the x-axis (see for instance [25, 79, 145, 146, 154] for Dyck paths and [18, 46, 72, 82] for Motzkin paths). In order to illustrate the effectiveness of generating functions as a counting tool, we give below two examples: for counting Dyck path and for Motzkin paths according to their length.

Any non-empty Dyck path $P \in \mathcal{D}$ has a unique first return decomposition [79] of the form $P = U\alpha D\beta$ where α and β are in turn two Dyck paths in $\mathcal{D}$. See Fig. 2.8 for an illustration of this decomposition.

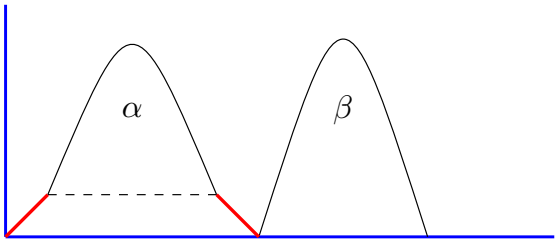


Figure 2.8: First return decomposition $U\alpha D\beta$ of a Dyck path $P \in \mathcal{D}$

Thus, a Dyck path $\boldsymbol{w} \in \mathcal{D}$ is either

- empty; the corresponding generating function is 1, or
- $\boldsymbol{w} = U\alpha D\beta$ where $\{\alpha, \beta\} \in \mathcal{D}$; the generating function for these words is $x^2 \cdot \mathcal{D}^2(x)$.

Combining this case, we deduce the functional equation

$$\mathcal{D}(x) = 1 + x^2 \cdot \mathcal{D}^2(x)$$

with the solution:

$$\mathcal{D}(x) = \frac{1 - \sqrt{1 - 4x^2}}{2x^2} = 1 + x^2 + 2x^4 + 5x^6 + 14x^8 + 42x^{10} + \mathbb{O}(x^{11}).$$

Now let $M \in \mathcal{M}$ be a non-empty Motzkin path, it can be uniquely written either as $M = U\alpha D\beta$ or $M = F\alpha$ where α and β are two Motzkin paths in $\mathcal{M}$. This decomposition will be called first return decomposition of $\mathcal{M}$ (see Fig. 2.9 for an illustration of this decomposition).

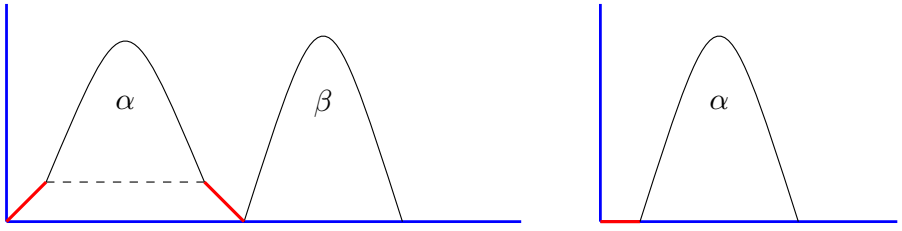


Figure 2.9: First return decomposition $U\alpha D\beta$ and $F\alpha$ of a Motzkin path $M \in \mathcal{M}$

A non-empty path $\boldsymbol{w} \in \mathcal{M}$ has one of the following forms:

- $\boldsymbol{w} = U\alpha D\beta$ where $\{\alpha, \beta\} \in \mathcal{M}$; the generating function for these paths is $x^2 \cdot \mathcal{M}^2(x)$,
- $\boldsymbol{w} = F\alpha$ where $\alpha \in \mathcal{M}$; the generating function for these paths is $x \cdot \mathcal{M}(x)$.

Combining these cases and considering the empty path which contributes with 1 to $\mathcal{M}(x)$, we deduce the functional equation for the paths in $\mathcal{M}$ according to their length

$$\mathcal{M}(x) = 1 + x \cdot \mathcal{M}(x) + x^2 \cdot \mathcal{M}^2(x).$$

Solving this equation, we will get the generating function:

$$\mathcal{M}(x) = \frac{1 - x - \sqrt{1 - 2x - 3x^2}}{2x^2}$$

and the first coefficients of $x^n, n \geq 0$, in the Taylor expansion are $1, 1, 2, 4, 9, 21, 51, 127$ (see Motzkin sequence A001006 in [213]).

2.3.4 Wilf equivalence

Two patterns π and τ are called *Wilf equivalent*, which is denoted $\pi \backsim \tau$, if the cardinality of the set of words avoiding π is equal to the cardinality of the set of words avoiding τ. Given two classes, $\mathcal{C}$ and $\mathcal{D}$, one natural question is to determine whether they are equinumerous, i.e. $|\mathcal{C}_n| = |\mathcal{D}_n|$ for every n. Two combinatorial classes that are equinumerous are said to be Wilf equivalent and the equivalence classes are called Wilf classes. For example, as is well known, both $\mathfrak{S}_n(123)$ and $\mathfrak{S}_n(132)$ are counted by the Catalan numbers, so all permutations of length three are in the same Wilf class.
Burstein [52] shows analytically that $123 \backsim 132$, while Burstein and Mansour [55] proved that $112 \backsim 121$, this allows us to obtain the Wilf equivalence classes for 3-patterns in words:

$$\{111\}, \{112, 121\}, \{123, 132\}$$

Of course, the more efficient the counting method is, the more cleverly it will rely on the properties of the elements in the finite set, therefore enumerative combinatorics is much more than just counting, but it has to do with understanding the structure of the objects we are counting.

2.4 Conclusion

This chapter was devoted to the presentation of introductory notions and bases in enumerative combinatorics. In particular, at first, we started by presenting different combinatorial objects to which this study relates (words, paths, permutation,...) and their statistics. Then, a brief explanation was presented on the different counting tools that we used in our contributions. This chapter therefore presents some definitions and results useful for a better understanding of the contributions of the following two chapters.

Chapter 3

Catalan words avoiding patterns

In this chapter, we present a contribution to a recent line of research on classical pattern avoidance on words subject to some growth restrictions (for instance, ascent sequences [30,91], inversion sequences [70,148,230], restricted growth functions [57,139]) by investigating connections between sequences on the On-line Encyclopedia of Integer Sequences [2] and Catalan words avoiding two patterns of length 3.

In this contribution, we recall Catalan words, which are words over the set of non-negative integers and we denote such words by sequences (for instance $w_1w_2\dots w_n$) or by italicized boldface letter (for instance $\boldsymbol{w}$ and $\boldsymbol{u}$). As defined in Section 2.1.1, the word $\boldsymbol{w} = w_1w_2\dots w_n$ is called a *Catalan word* if

$$w_1 = 0 \text{ and } 0 \leq w_i \leq w_{i-1} + 1 \text{ for } i = 2, 3, \dots, n.$$

We remind that $\mathcal{C}_n$ is the set of length n Catalan words and $c_n = |\mathcal{C}_n|$ is the nth Catalan number $\frac{1}{n+1}\binom{2n}{n}$. For a pattern π, we denote by $\mathcal{C}_n(\pi)$ the set of length n Catalan words avoiding π, and $c_n(\pi) = |\mathcal{C}_n(\pi)|$ is the cardinality of $\mathcal{C}_n(\pi)$ and $\mathcal{C}(\pi) = \cup_{n\geq 0}\mathcal{C}_n(\pi)$. For example, $\mathcal{C}_n(101)$ is the set of length n Catalan words avoiding 101, that is, the set of words $\boldsymbol{w}$ in $\mathcal{C}_n$ such that there are no i, j and k, $1 \leq i < j < k \leq n$, with $w_i = w_k > w_j$. So, $\mathcal{C}_4(101) = \{0000, 0001, 0010, 0011, 0012, 0100, 0110, 0111, 0112, 0120, 0121, 0122, 0123\}$.

Likewise, if π is the set of patterns $\{\alpha, \beta, \dots\}$, then $\mathcal{C}_n(\pi)$ and $\mathcal{C}_n(\alpha, \beta, \dots)$ denote both the set of length n Catalan words avoiding each pattern in π; and $c_n(\pi) = c_n(\alpha, \beta, \dots)$ and $\mathcal{C}(\pi) = \mathcal{C}(\alpha, \beta, \dots)$ have similar meaning as above. A *descent* in a word $\boldsymbol{w} = w_1w_2\dots w_n$ is a position i, $1 \leq i \leq n-1$, with $w_i > w_{i+1}$. The (ordinary) generating function of a set of pattern avoiding Catalan words $\mathcal{C}(\pi)$ is the formal power series

$$C_\pi(x) = \sum_{n\geq 0} c_n(\pi)x^n = \sum_{\boldsymbol{w}\in\mathcal{C}(\pi)} x^{|\boldsymbol{w}|},$$

where $|\boldsymbol{w}|$ is the length of the word $\boldsymbol{w}$. In our case of generating function approach for counting classes of pattern avoiding Catalan words we consider the descent number as an additional statistic obtaining 'for free' the bivariate generating function

$$C_\pi(x, y) = \sum_{\boldsymbol{w}\in\mathcal{C}(\pi)} x^{|\boldsymbol{w}|}y^{\mathrm{des}(\boldsymbol{w})},$$

where $\mathrm{des}(\boldsymbol{w})$ is the number of descents of $\boldsymbol{w}$. With these notations, the coefficient of $x^n y^k$ in $C_\pi(x, y)$ is the number of Catalan words of length n avoiding π and having k descents, and for a set $\mathcal{S}$ of Catalan words $S(x)$ and $S(x, y)$ have a similar meaning.

For a word $\boldsymbol{w} = w_1 w_2 \dots w_n$ and an integer a, we denote by $(\boldsymbol{w} + a)$ the word obtained from $\boldsymbol{w}$ by increasing by a each of its entries, that is, the word $(w_1 + a)(w_2 + a) \cdots (w_n + a)$. In our constructions, we will often make use of two particular families of Catalan words: those avoiding 10 (i.e., with no descents) and we call these words *weakly increasing* (or *w.i* for short) *Catalan words*; and those avoiding 00 (and thus necessarily avoiding 10) and we call these words *strictly increasing* (or *s.i* for short) *Catalan words*. It is easy to see that for each length $n \geq 1$ there are 2^{n-1} w.i. Catalan words and one s.i. Catalan word.

There is many studies on Catalan words [29,209], for example, Baril, Kirgizov and Vajnovszki [23] studied the distribution of descents on the sets of Catalan words avoiding a pattern of length at most three, and Keith [228] proved combinatorially a conjecture of Chunwei Song on a limiting case of the q, t-Schröder theorem where the proof matches pairs of tableaux to Catalan words in a manner that preserves differences in the maj statistic.

The remaining of this chapter is structured as follows. In the next subsection, we characterize pattern avoiding ascent sequences which are Catalan words, establishing ties with some similar enumerative results for ascent sequences in [30]. In Section 3.2 we consider classes of Catalan words avoiding both a length two and a length three pattern. In the next sections we discuss Catalan words avoiding two patterns of length three, in increasing order of their complexity: obvious cases (Section 3.3), cases counted via recurrences (Section 3.4) and cases counted via generating functions (Section 3.5); these results are summarized in Table 3.2. We conclude with some remarks and further research directions.

3.1 Catalan words vs. ascent sequences

An *ascent* in a word $\boldsymbol{w} = w_1 w_2 \dots w_n$ is a position i, $1 \leq i \leq n-1$, with $w_i < w_{i+1}$, and $\mathrm{asc}(\boldsymbol{w})$ denotes the number of ascents in $\boldsymbol{w}$. Closely related to Catalan words are ascent sequences introduced in [43] and defined as: the word $\boldsymbol{w} = w_1 w_2 \dots w_n$ is called an *ascent sequence* if

$$w_1 = 0 \text{ and } 0 \leq w_i \leq \mathrm{asc}(w_1 w_2 \dots w_{i-1}) + 1 \text{ for } i = 2, 3, \dots, n,$$

and $\mathcal{A}_n$ denotes the set of length n ascent sequences, and $\mathcal{A} = \cup_{n \geq 0} \mathcal{A}_n$. Similarly as for Catalan words, if π is a pattern, then $\mathcal{A}_n(\pi)$ is the set of length n ascent sequences avoiding π, and $\mathcal{A}(\pi) = \cup_{n \geq 0} \mathcal{A}_n(\pi)$. Clearly, $\mathcal{C}_n = \mathcal{A}_n$ for $n \leq 3$, and $\mathcal{C}_n \subset \mathcal{A}_n$ for $n \geq 4$, and this inclusion is strict, for instance, $0102 \in \mathcal{A}_4 \setminus \mathcal{C}_4$. It turns out that, for particular patterns π, $\mathcal{A}_n(\pi)$ collapses to $\mathcal{C}_n(\pi)$ for any n, and this behavior where the pattern 0102 plays a critical role is discussed below.

Proposition 1. *If $\boldsymbol{w} \in \mathcal{A} \setminus \mathcal{C}$, then $\boldsymbol{w}$ contains the pattern* 0102.

Proof. If $\boldsymbol{w} = w_1 w_2 \dots w_n$ is an ascent sequence which is not a Catalan word, then there is an i such that $w_i \geq w_{i-1} + 2$, and let k be the smallest such i. It follows that $w_i \leq w_{i-1} + 1$ for any i, $2 \leq i \leq k-1$, or equivalently $w_1 w_2 \dots w_{k-1}$ is a Catalan word. Thus, if $w_i > 0$, $2 \leq i \leq k-1$, then each symbol less than w_i occurs in the prefix $w_1 w_2 \dots w_{i-1}$. We distinguish two cases: (i) w_{k-1} is not the maximal symbol of the prefix $w_1 w_2 \dots w_{k-1}$, and (ii) otherwise.

(i) In this case, there exist i and j, $1 \leq i < j < k-1$, such that $w_j = w_{k-1} + 1$ and $w_i = w_{k-1}$. It follows that $w_i w_j w_{k-1} w_k$ is an occurrence of 0102.

(ii) In this case the prefix $w_1w_2\dots w_{k-1}$ has a descent (otherwise, since $\boldsymbol{w}$ is an ascent sequence, the maximal possible value for w_k is $w_{k-1}+1$), and let j be such a descent, that is $w_j > w_{j+1}$, $j+1 < k-1$. As noticed above, the symbol w_{j+1} already occurs in $w_1w_2\dots w_{j-1}$, say in position i. Thus, $w_iw_jw_{j+1}w_k$ is an occurrence of 0102. □

Proposition 2. *For $n \geq 4$ and a pattern π the followings are equivalent*

1. $\mathcal{A}_n(\pi) = \mathcal{C}_n(\pi)$,
2. 0102 *contains the pattern* π.

Proof. '2. $\Rightarrow$ 1.' We proceed by contraposition and considering $\mathcal{C}_n(\pi) \subseteq \mathcal{A}_n(\pi)$. If $\boldsymbol{w} \in \mathcal{A}_n(\pi) \setminus \mathcal{C}_n(\pi)$, then, by Proposition 1, $\boldsymbol{w}$ contains 0102, and so 0102 does not contain π.

'1. $\Rightarrow$ 2.' Again by contraposition: if 0102 does not contain π, then at least one of the words $01023\cdots(n-2)$ or 01020^{n-4} belongs to $\mathcal{A}_n(\pi)$ and not to $\mathcal{C}_n(\pi)$, and so $\mathcal{A}_n(\pi) \neq \mathcal{C}_n(\pi)$. □

Since the only patterns of length three of 0102 are 001, 010, 012 and 102, we have the following consequence of Proposition 2.

Corollary 1. *For $n \geq 4$ and a pattern π of length three, $\mathcal{A}_n(\pi) = \mathcal{C}_n(\pi)$ if and only if $\pi \in \{001, 010, 012, 102\}$.*

Pattern avoidance in ascent sequences was initiated in [91], and in [30] ascent sequences avoiding a pair of patterns of length three are considered and exact enumeration for several such pairs is given. In light of Corollary 1 it can happen that if a pattern of the avoided pair is one of the four specified in this corollary, then the resulting ascent sequences are Catalan words as well. The pairs of avoided patterns for which ascent sequences and Catalan words coincide, and for which the enumeration has already been considered in [30] are highlighted in the summarizing Table 3.2. In order to keep the present article self-contained we fully consider these cases, our proofs being alternative to those in [30].

3.2 Avoiding a length two and a length three pattern

There are three patterns of length two, namely 00, 01 and 10, and we have:

Proposition 3. *The number of Catalan words avoiding a pattern of length two and a pattern π of length three is given by:*

$$c_n(00,\pi) = \begin{cases} 0 & \text{if } \pi = 012 \text{ and } n \geq 3, \\ 1 & \text{elsewhere;} \end{cases}$$

$$c_n(01,\pi) = \begin{cases} 0 & \text{if } \pi = 000 \text{ and } n \geq 3, \\ 1 & \text{elsewhere;} \end{cases}$$

$$c_n(10,\pi) = \begin{cases} F_n & \text{if } \pi = 000, \\ n & \text{if } \pi \in \{001, 011, 012\}, \\ 2^{n-1} & \text{elsewhere,} \end{cases}$$

*where F_n is the n*th *Fibonacci number defined by $F_0 = F_1 = 1$ and for $n \geq 2$*

$$F_n = F_{n-1} + F_{n-2}$$

Proof. If $\boldsymbol{w} \in \mathcal{C}_n(00)$, then $\boldsymbol{w} = 012\cdots(n-1)$. Thus, $\mathcal{C}_n(00,\pi) = \{012\cdots(n-1)\}$ except when $\pi = 012$ and in this case $\mathcal{C}_n(00,\pi) = \varnothing$ for $n \geq 3$, and the counting relation for $\mathcal{C}_n(00,\pi)$ follows.
Similarly, if $\boldsymbol{w} \in \mathcal{C}_n(01)$, then $\boldsymbol{w} = 00\cdots0$. Thus, $\mathcal{C}_n(01,\pi) = \{000\cdots0\}$ except when $\pi = 000$ and in this case $\mathcal{C}_n(01,\pi) = \varnothing$ for $n \geq 3$.

Finally, a Catalan word avoids 10 if and only if it avoids 010. It follows that $c_n(10,\pi) = c_n(010,\pi)$, which falls in the case of avoidance of two length 3 patterns and the corresponding proofs are given in the next section, see also Table 3.2. □

3.3 Trivial cases

Superfluous patterns

If the pattern τ contains the pattern σ, then clearly $\mathcal{C}_n(\sigma,\tau) = \mathcal{C}_n(\sigma)$; but this phenomenon can occur even when σ and τ are not related by containment and in this case, following [30], we say that τ is a *superfluous pattern* for σ. For example, any word in $\mathcal{C}_n(012)$ is a binary word, and thus any pattern with at least three different symbols is a superfluous pattern for 012. In Table 3.1 are listed all pairs of superfluous patterns of length three. It is worth to mentioning that superfluousness is a transitive relation, for instance, 201 is superfluous for 021 which in turn is superfluous for 011. So, a pattern can be superfluous for several other ones, for instance $\tau = 201$ is superfluous for each of the patterns $001, 010, 011, 012, 021, 101, 120$. Also, it is easy to see that if τ is superfluous for σ, then τ is larger lexicographically than σ.

σ	000	001	010	011	012	021	101	110	120
	100	101	021	021	021	201	102	210	201
		102	100	101	102	210	201		210
		201	101	102	120				
τ			102	110	201				
			110	201	210				
			120	210					
			201						
			210						

Table 3.1: In each column, each pattern τ is superfluous for the pattern σ.

Ultimately constant sequences

It can happen that the number of Catalan words avoiding a pair of length 3 patterns is constant for enough long words. The only two such cases are given below.

Proposition 4.

$$c_n(000,011) = \begin{cases} 1 & \text{if } n = 1, \\ 2 & \text{if } n = 2, \\ 3 & \text{if } n \geq 3; \end{cases}$$

and

$$c_n(000,012) = \begin{cases} 1 & \text{if } n = 1, \\ 2 & \text{if } n = 2, \\ 3 & \text{if } n = 3 \text{ or } n = 4, \\ 0 & \text{if } n \geq 5. \end{cases}$$

Proof. If $n \geq 3$, then $\mathcal{C}_n(000,011) = \{0\boldsymbol{x}, \boldsymbol{x}0, \boldsymbol{x}(n-1)\}$ where $\boldsymbol{x}$ is the word $01\cdots(n-2)$, and the first point follows.
If a Catalan word avoids 012, then it is a binary word. In addition, if its length is larger than 4 it necessarily contains three identical entries, and so $\mathcal{C}_n(000,012) = \varnothing$ for $n \geq 5$. Considering the initial values of $c_n(000,012)$ the second point follows.

□

Counting sequence n

Proposition 5. *If π is one of the pairs of patterns $\{001,010\}$, $\{001,011\}$, $\{001,012\}$, $\{010,011\}$, $\{010,012\}$ or $\{011,012\}$, then $c_n(\pi) = n$.*

Proof. The proof is based on the easy to understand description given below for the corresponding sets of Catalan words:

π	$\mathcal{C}_n(\pi)$	π	$\mathcal{C}_n(\pi)$
$\{001,010\}$	$\{012\cdots jj\cdots j \,:\, 0 \leq j \leq n-1\}$	$\{010,011\}$	$\{0^j12\cdots(n-j) \,:\, 1 \leq j \leq n\}$
$\{001,011\}$	$\{012\cdots j0\cdots0 \,:\, 0 \leq j \leq n-1\}$	$\{010,012\}$	$\{0^{j+1}1^{n-j-1} \,:\, 0 \leq j \leq n-1\}$
$\{001,012\}$	$\{01^j0^{n-j-1} \,:\, 0 \leq j \leq n-1\}$	$\{011,012\}$	$\{0^n\} \cup \{0^j10^{n-j-1} \,:\, 1 \leq j \leq n-1\}$

□

3.4 Counting via recurrence

Counting sequence $2(n-1)$

Proposition 6. *If $\pi = \{011,100\}$ or $\pi = \{011,120\}$, then*

$$c_n(\pi) = 2(n-1)$$

for $n \geq 2$ (Sequence A005843 in [2]).

Proof. If $\pi = \{011,100\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, $n \geq 2$, then either

- $\boldsymbol{w} = 0^k\boldsymbol{u}$, $0 \leq k \leq n-1$, with $\boldsymbol{u}$ a s.i. Catalan word (of length at least one), or
- $\boldsymbol{w} = 0^k\boldsymbol{u}0$, $0 \leq k \leq n-3$, with $\boldsymbol{u}$ a s.i. Catalan word (of length at least two).

In the first case, there are n possibilities for $\boldsymbol{w}$ and $n-2$ possibilities in the second case, and the result holds. If $\pi = \{011,120\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, $n \geq 2$, then either

- $\boldsymbol{w} = 0^k\boldsymbol{u}$, $0 \leq k \leq n-1$, with $\boldsymbol{u}$ a s.i. Catalan word, or
- $\boldsymbol{w} = 0^k 10^{n-k-1}$, $1 \leq k \leq n-2$,

and as previously the result holds. □

Sequences involving 2^n

Proposition 7. *If $\pi = \{000, 101\}$, then*

$$c_n(\pi) = 2^{n-1}$$

for $n \geq 1$ (A000079 in [2]).

Proof. If a Catalan word avoids 101, then it is unimodal (that is, it can be written not necessarily in a unique way as $\boldsymbol{uv}$ with $\boldsymbol{u}$ a weakly decreasing and $\boldsymbol{v}$ weakly decreasing word). In addition, if the word avoids 000, then its maximal value occurs at most twice, and when it occurs twice this happens in consecutive positions.
We denote by $\mathcal{D}_n$ the subset of words in $\mathcal{C}_n(\pi)$ where the maximal entry occurs once and by $\mathcal{E}_n$ that where it occurs twice, $d_n = |\mathcal{D}_n|$, $e_n = |\mathcal{E}_n|$, and $c_n(\pi) = d_n + e_n$. Any word $\boldsymbol{w} = w_1 \cdots mm \cdots w_{n-1} \in \mathcal{E}_{n-1}$ with its maximal value m occurring twice can be extended into a word in $\mathcal{D}_n$ by one of the transformations:

$$\boldsymbol{w} \mapsto w_1 \cdots m(m+1)m \cdots w_{n-1}, \text{ and}$$

$$\boldsymbol{w} \mapsto w_1 \cdots mm(m+1) \cdots w_{n-1},$$

and any word $\boldsymbol{w} = w_1 \cdots m \cdots w_{n-1} \in \mathcal{D}_{n-1}$ with its maximal value m occurring once can be extended into a word in $\mathcal{D}_n$ by:

$$\boldsymbol{w} \mapsto w_1 \cdots m(m+1) \cdots w_{n-1}.$$

Conversely, any word in $\mathcal{D}_n$, $n \geq 2$, can uniquely be obtained from a word in $\mathcal{D}_{n-1}$ or in $\mathcal{E}_{n-1}$ by reversing one of the transformations above, so

$$d_n = 2 \cdot e_{n-1} + d_{n-1},$$

for $n \geq 2$.
Reasoning in a similar way we have

$$e_n = 2 \cdot e_{n-2} + d_{n-2},$$

Thus, for $n \geq 3$, $e_n = d_{n-1}$, and finally

$$c_n(\pi) = d_n + e_n = 2 \cdot (d_{n-1} + e_{n-1}) = 2 \cdot c_{n-1}(\pi),$$

and with the initial conditions $c_1(\pi) = 1$ and $c_2(\pi) = 2$, the result follows. □

Proposition 8. *If $\pi = \{101, 210\}$ or $\pi = \{102, 120\}$, then*

$$c_n(\pi) = (n-1) \cdot 2^{n-2} + 1$$

for $n \geq 2$ (A005183 in [2]).

Proof. If $\pi = \{101, 210\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, $n \geq 3$, then either

- $\boldsymbol{w} = 0\boldsymbol{u}$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or
- $\boldsymbol{w} = 0(\boldsymbol{u}+1)$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or
- $\boldsymbol{w} = 0(\boldsymbol{u}+1)0^{n-k-1}$ with $\boldsymbol{u}$ a w.i. Catalan word of length k, $1 \leq k \leq n-2$.

The number of words in each of the first two cases is $c_{n-1}(\pi)$. The number of length k w.i. Catalan words is 2^{k-1}, so the number of words in the last case is

$$\sum_{k=1}^{n-2} 2^{k-1} = \sum_{k=0}^{n-3} 2^k = 2^{n-2} - 1.$$

Thus, $c_n(\pi) = 2c_{n-1}(\pi) + 2^{n-2} - 1$, and after calculation the result holds.

If $\pi = \{102, 120\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, $n \geq 3$, then either

- $\boldsymbol{w} = 0\boldsymbol{u}$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or
- $\boldsymbol{w} = 0(\boldsymbol{u}+1)$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or
- $\boldsymbol{w} = 01\boldsymbol{u}$ with $\boldsymbol{u}$ a length $(n-2)$ binary word other than $11\cdots1$.

The number of words in each of the first two cases is $c_{n-1}(\pi)$, and the number of words in the last case is $2^{n-2} - 1$. So $c_n(\pi) = 2c_{n-1}(\pi) + 2^{n-2} - 1$, and again the result holds. □

Proposition 9. *If π is one of the pairs of patterns $\{021, 100\}$, $\{021, 101\}$, $\{101, 110\}$ or $\{101, 120\}$, then*

$$c_n(\pi) = 2^n - n$$

for $n \geq 0$ (A000325 in [2]).

Proof. If $\pi = \{021, 100\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, $n \geq 4$, then either

- $\boldsymbol{w}$ is a w.i. Catalan word, or
- $\boldsymbol{w} = \boldsymbol{u}0$ with $\boldsymbol{u}$ a w.i. Catalan word of length $(n-1)$ other than $00\cdots0$, or
- $\boldsymbol{w} = 0\boldsymbol{v}0(\boldsymbol{u}+1)$ where $\boldsymbol{v}$ is w.i. binary word ending by 1 and $\boldsymbol{u}$ is a w.i. Catalan word of length k, $1 \leq k \leq n-3$.

In the first case the number of words $\boldsymbol{w}$ is 2^{n-1} and in the second case the number of words $\boldsymbol{w}$ is $2^{n-2} - 1$. In the last, case the number of words $\boldsymbol{w}$ is

$$\sum_{k=1}^{n-3} (n-k-2) \cdot 2^{k-1} = 2^{n-2} - (n-1).$$

Combining these cases and considering the initial values of $c_n(\pi)$ the result holds.

If $\pi = \{021, 101\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, $n \geq 2$, then either

- $\boldsymbol{w} = \boldsymbol{u}0$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or
- $\boldsymbol{w} = 0^{n-k}(\boldsymbol{u}+1)$ with $\boldsymbol{u}$ a w.i. Catalan word of length k, $1 \leq k \leq n-1$.

The number of words in the first case is $c_{n-1}(\pi)$ and the number of words in the second case is $\sum_{k=1}^{n-1} 2^{k-1} = 2^{n-1} - 1$. So $c_n(\pi) = c_{n-1}(\pi) + 2^{n-1} - 1$ and after calculation the result follows.

If $\pi = \{101, 110\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, $n \geq 3$, then either

- $\boldsymbol{w} = 0\boldsymbol{u}$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or
- $\boldsymbol{w} = 0(\boldsymbol{u}+1)$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or
- $\boldsymbol{w} = \boldsymbol{u}0^{n-k}$ with $\boldsymbol{u}$ a s.i. Catalan word of length k, $2 \leq k \leq n-1$.

The number of words $\boldsymbol{w}$ in each of the first two cases is $c_{n-1}(\pi)$. For the last case, for each k there is exactly one word $\boldsymbol{w}$, so their number is $(n-2)$ in this case. So $c_n(\pi) = 2c_{n-1}(\pi) + n - 2$ and after calculation the result holds.

If $\pi = \{101, 120\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, $n \geq 3$, then either

- $\boldsymbol{w} = 0\boldsymbol{u}$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or
- $\boldsymbol{w} = 0(\boldsymbol{u}+1)$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or
- $\boldsymbol{w} = 01^k0^{n-k-1}$, $1 \leq k \leq n-2$.

So, again $c_n(\pi) = 2c_{n-1}(\pi) + n - 2$.

□

Proposition 10. *If $\pi = \{021, 120\}$, then*

$$c_n(\pi) = (n+2) \cdot 2^{n-3}$$

for $n \geq 3$ (A045623 in [2]).

Proof. If $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, $n \geq 4$, then either

- $\boldsymbol{w} = 0\boldsymbol{u}0$ with $\boldsymbol{u}$ a length $(n-2)$ binary word, or
- $\boldsymbol{w} = 0(\boldsymbol{u}+1)$ with $\boldsymbol{u}$ a length $(n-1)$ w.i. Catalan word, or
- $\boldsymbol{w} = 0\boldsymbol{u}0(\boldsymbol{v}+1)$ with $\boldsymbol{u}$ a binary word and $\boldsymbol{v}$ w.i. Catalan word.

The number of words in each of the first two cases is 2^{n-2} and the number of words in the last case is

$$\sum_{k=1}^{n-2} 2^{n-k-2}2^{k-1} = (n-2)2^{n-3},$$

and so $c_n(\pi) = 2^{n-1} + (n-2)2^{n-3}$, which gives the desired result. □

If a Catalan word avoids both 102 and 110, then it has at most one descent. In the second part of the proof of the next proposition we need the following technical lemma which gives the number of Catalan words in $\mathcal{C}_n(102, 110)$ with one descent and avoiding the pattern 00 before the descent (note that in this case avoiding 00 is equivalent to avoiding equal consecutive entries). The set of these words is empty for $n \leq 2$, it is the single word set $\{010\}$ for $n = 3$, and $\{0100, 0101, 0120, 0121\}$ for $n = 4$.

Lemma 1. *Let $\mathcal{D}_n$ be the set of words in $\mathcal{C}_n(102, 110)$ having one descent and avoiding 00 before the descent. Then $|\mathcal{D}_n| = \frac{n}{6}(n-1)(n-2)$.*

Proof. A word belongs to $\mathcal{D}_n$, $n \geq 3$, if and only if it can be written as

$$012\ldots(k-1)a(\boldsymbol{u}+a),$$

with $2 \leq k \leq n-1$ (k is the position of the unique descent in the word), $a \in \{0, 1, \ldots k-2\}$, and $\boldsymbol{u}$ is a length $(n-k-1)$ w.i. Catalan word over $\{0, 1\}$. For each choice of k, there are $k-1$ choices for a, and for each choice for a there are $n-k$ choices for $\boldsymbol{u}$. It follows that

$$|\mathcal{D}_n| = \sum_{k=2}^{n-1}(k-1)\cdot(n-k),$$

and after calculation the statement holds. □

Proposition 11. *If $\pi = \{021, 102\}$ or $\pi = \{102, 110\}$, then*

$$c_n(\pi) = 3\cdot 2^{n-1} - \frac{1}{2}(n+1)(n+2) + n$$

for $n \geq 1$ (A116702 in [2]).

Proof. If $\pi = \{021, 102\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, $n \geq 3$, then either

- $\boldsymbol{w} = 0\boldsymbol{u}$ with $\boldsymbol{u}$ a length $(n-1)$ binary word, or
- $\boldsymbol{w} = 0\cdots0(\boldsymbol{u}+1)0\cdots0$ with $\boldsymbol{u}$ a length k, $2 \leq k \leq n-1$, w.i. Catalan word other than $00\cdots0$ and $\boldsymbol{w}$ beginning by at least one 0.

The number of words in the first case is 2^{n-1} and the number of those in the second case is

$$\begin{aligned}&\sum_{k=2}^{n-1}(2^{k-1}-1)\cdot(n-k)\\ &= 2\cdot 2^{n-1} - \frac{1}{2}(n+1)(n+2) + n,\end{aligned}$$

and combining the two cases the result holds.

If $\pi = \{102, 110\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, then either

- $\boldsymbol{w}$ is a w.i. Catalan word, the number of such words is 2^{n-1}, or
- $\boldsymbol{w} \in \mathcal{D}_n$, with $\mathcal{D}_n$ defined in Lemma 1, or
- $\boldsymbol{w} = \boldsymbol{u}(\boldsymbol{v}+m)$, where $\boldsymbol{u}$ is a w.i. Catalan word of length k, $1 \leq k \leq n-3$, m is the maximal (rightmost) entry of $\boldsymbol{u}$, and $\boldsymbol{v}$ is a word belonging to $\mathcal{D}_{n-k}$.

Indeed, the first case corresponds to words with no descents, the second one to those with a descent and no occurrences of 00 before the descent, and the third one to those with both descent and occurrences of 00 before the descent (the rightmost such occurrence is in positions k and $k+1$). By Lemma 1, the number of words in the third case is

$$\sum_{k=1}^{n-3} 2^{k-1} \cdot (n-k) \cdot \left(\frac{1}{6}(n-k)^2 - \frac{1}{2}(n-k) + \frac{1}{3} \right)$$
$$= 2^n - \frac{1}{6}(n+1)(n^2-n+6).$$

Finally, combining the three previous cases the desired result holds. □

Sequences involving binomial coefficients

In this part, we use the notation

$$ab \cdots \underset{\substack{\uparrow \\ i}}{cde} \cdots f$$

to denote that the entry d is in position i in the word $ab \cdots cde \cdots f$.

Proposition 12. *If $\pi = \{001, 210\}$, then*

$$c_n(\pi) = \binom{n}{3} + n$$

for $n \geq 3$ (A000125 in [2]).

Proof. If $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, then it has at most one descent.

If $\boldsymbol{w}$ has no descents, then it has the form $\boldsymbol{w} = 01 \cdots (m-1)m \cdots m$ and there are n such words.

If $\boldsymbol{w}$ has one descent, then it has the form $\boldsymbol{w} = 01 \cdots (m-1)m \cdots mk \cdots k$ with $0 \leq k < m < n-1$. It follows that there is a bijection between the family of 3-element subsets of $\{0, 1, \ldots, n-1\}$ and the words in $\mathcal{C}_n(\pi)$ with one descent:

$$\{k, m, \ell\} \mapsto 0123 \cdots (m-1)m \cdots \underset{\substack{\uparrow \\ \ell}}{mk} \cdots k.$$

Combining the two cases, we have $c_n(\pi) = \binom{n}{3} + n$. □

Proposition 13. *If π is one of the pairs of patterns $\{001, 021\}$, $\{001, 110\}$, $\{001, 120\}$, $\{012, 100\}$, $\{012, 101\}$ or $\{012, 110\}$, then*

$$c_n(\pi) = \binom{n}{2} + 1$$

for $n \geq 2$ (A000124 in [2]).

Proof. In any of the six cases for π, the set $\mathcal{C}_n(\pi)$ is in bijection with the family S of subsets of $\{2, \ldots, n\}$ with at most two elements. We give below explicit definitions for such bijections, where the empty set is mapped to $0 \cdots 0 \in \mathcal{C}_n(\pi)$ by each of them.

If $\pi = \{001, 021\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, then either $\boldsymbol{w} = 00 \cdots 0$ or for some $m \geq 1$, $\boldsymbol{w} = 0123 \cdots (m-1)m^s$ with $s \geq 1$ or $\boldsymbol{w} = 0123 \cdots (m-1)m^s 0^t$ with $s \geq 1$, $t \geq 1$, and the desired bijection $S \to \mathcal{C}_n(001, 021)$ is

$$\{k\} \mapsto 0123\cdots(m-1)\underset{\substack{\uparrow\\ k=m+1}}{m}\cdots m;$$
$$\{k,j\} \mapsto 0123\cdots(m-1)\underset{\substack{\uparrow\\ k=m+1}}{m}\cdots m0\cdots\underset{\substack{\uparrow\\ j}}{0}.$$

If $\pi = \{001, 110\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, then either $\boldsymbol{w} = 00\cdots0$ or $\boldsymbol{w} = 0123\cdots(m-1)m\cdots m$ with m the maximal entry of $\boldsymbol{w}$, or $\boldsymbol{w} = 0123\cdots(m-1)m\ell\cdots\ell$ with $\ell < m$, and the desired bijection $S \to \mathcal{C}_n(001, 110)$ is

$$\{k\} \mapsto 0123\cdots(m-1)\underset{\substack{\uparrow\\ k=m+1}}{m}\cdots m;$$
$$\{k,j\} \mapsto 0123\cdots(j-2)(k-2)\cdots(k-2).$$

If $\pi = \{001, 120\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, then either $\boldsymbol{w} = 00\cdots0$, or for some $m \geq 1$, $\boldsymbol{w} = 0123\cdots(m-1)m^s$ with $s \geq 1$ or $\boldsymbol{w} = 0123\cdots(m-1)m^s(m-1)^t$ with $s, t \geq 1$, and the desired bijection $S \to \mathcal{C}_n(001, 120)$ is

$$\{k\} \mapsto 0123\cdots(m-1)\underset{\substack{\uparrow\\ k=m+1}}{m}\cdots m;$$
$$\{k,j\} \mapsto 0123\cdots(m-1)\underset{\substack{\uparrow\\ k=m+1}}{m}\cdots m(m-1)\cdots\underset{\substack{\uparrow\\ j}}{(m-1)}.$$

For the next three cases we need the following observation: a Catalan word avoids 012 if and only if it is a binary word (over $\{0, 1\}$) beginning by a 0.

If $\pi = \{012, 100\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, then either $\boldsymbol{w} = 0^s1^t$ with $s \geq 1$, $t \geq 0$ or $\boldsymbol{w} = 0^s1^t01^r$ with $s, t \geq 1$, $r \geq 0$, and the desired bijection $S \to \mathcal{C}_n(012, 100)$ is

$$\{k\} \mapsto 0\cdots\underset{\substack{\uparrow\\ k}}{0}1\cdots1;$$
$$\{k,j\} \mapsto 0\cdots\underset{\substack{\uparrow\\ k}}{0}1\cdots\underset{\substack{\uparrow\\ j}}{1}01\cdots1.$$

If $\pi = \{012, 101\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, then either $\boldsymbol{w} = 0^s1^t$ with $s \geq 1$, $t \geq 0$ or $\boldsymbol{w} = 0^s1^t0^r$ with $s, t \geq 1$, $r \geq 1$, and the desired bijection $S \to \mathcal{C}_n(012, 101)$ is

$$\{k\} \mapsto 0\cdots\underset{\substack{\uparrow\\ k}}{0}1\cdots1;$$
$$\{k,j\} \mapsto 0\cdots\underset{\substack{\uparrow\\ k}}{0}1\cdots\underset{\substack{\uparrow\\ j}}{1}0\cdots0.$$

If $\pi = \{012, 110\}$ and $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, then either $\boldsymbol{w} = 0^s1^t$ with $s \geq 1$, $t \geq 0$ or $\boldsymbol{w} = 0^s10^t1^r$ with $s, t \geq 1$, $r \geq 0$, and the desired bijection $S \to \mathcal{C}_n(012, 110)$ is

$$\{k\} \mapsto 0\cdots\underset{\substack{\uparrow\\ k}}{0}1\cdots1;$$
$$\{k,j\} \mapsto 0\cdots\underset{\substack{\uparrow\\ k}}{0}10\cdots\underset{\substack{\uparrow\\ j}}{0}1\cdots1.$$

□

Sequences involving Fibonacci(-like) numbers

As in Proposition 3, we consider the sequence of Fibonacci numbers $(F_n)_{n\geq0}$ defined as $F_0 = 1$, $F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$.

Proposition 14. *If* $\pi = \{000, 001\}$ *or* $\pi = \{000, 010\}$*, then*

$$c_n(\pi) = F_n$$

for $n \geq 0$ *(*A00045 *in [2]).*

Proof. For $\pi = \{000, 001\}$ the proof is up to a certain point similar to that of Proposition 7. A word belonging to $\mathcal{C}_n(\pi)$ is unimodal and its maximal entry occurs once or twice in consecutive positions. Let $\mathcal{D}_n$ denote the subset of words in $\mathcal{C}_n(\pi)$ where the maximal entry occurs once and $\mathcal{E}_n$ denote that where it occurs twice. If $\boldsymbol{w} \in \mathcal{C}_n(\pi)$ has its maximal entry m, then the insertion of $(m+1)$ after the leftmost occurrence of m in $\boldsymbol{w}$ produces a word in $\mathcal{D}_{n+1}$, and the insertion of $(m+1)(m+1)$ produces a word in $\mathcal{E}_{n+2}$. It is easy to see that these transformations induce a bijection between $\mathcal{C}_n(\pi)$ and $\mathcal{D}_{n+1}$, and between $\mathcal{C}_n(\pi)$ and $\mathcal{E}_{n+2}$, and thus between $\mathcal{C}_{n-2}(\pi) \cup \mathcal{C}_{n-1}(\pi)$ and $\mathcal{C}_n(\pi)$. It follows that $c_n(\pi)$ satisfies a Fibonacci-like recurrence, and by considering the initial values for $c_n(\pi)$ the result holds.

For $\pi = \{000, 010\}$, a word $\boldsymbol{w} \in C_n(\pi)$ is characterized by: $\boldsymbol{w}$ is w.i. and $\boldsymbol{w}$ does not have three consecutive equal entries. So $\boldsymbol{w}$ can be represented by the binary word $b_1 b_2 \dots b_{n-1}$ with no two consecutive 1s where $b_i = 1$ iff $w_i = w_{i-1}$. This representation is a bijection between $\mathcal{C}_n(\pi)$ and the set of binary words of length $(n-1)$ without two consecutive 1s, which cardinality is the Fibonacci number, see for instance [216]. □

Proposition 15. *If* $\pi = \{001, 100\}$*, then*

$$c_n(\pi) = F_{n+1} - 1$$

for $n \geq 1$.

Proof. If $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, $n \geq 3$, then either

- $\boldsymbol{w} = 0 \cdots 0$, or
- $\boldsymbol{w} = 0(\boldsymbol{u}+1)$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or
- $\boldsymbol{w} = 0(\boldsymbol{u}+1)0$ with $\boldsymbol{u} \in \mathcal{C}_{n-2}(\pi)$.

So, $c_n(\pi)$ satisfies the recurrence $c_n(\pi) = c_{n-1}(\pi) + c_{n-2}(\pi) + 1$ for $n \geq 3$, and solving it we have the desired result. □

In the next proposition, we will make use of the following relation satisfied by the even index Fibonacci numbers: $F_{2n} = F_{2n-2} + \sum_{i=0}^{n-1} F_{2i}$ for $n \geq 1$.

Proposition 16. *If* $\pi = \{100, 201\}$*, then*

$$c_n(\pi) = F_{2n-2}$$

for $n \geq 1$ *(*A001519 *in [2]).*

Proof. If $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, $n \geq 3$, then either

- $\boldsymbol{w} = 0\boldsymbol{u}$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or
- $\boldsymbol{w} = 0(\boldsymbol{u}+1)$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or

- $\boldsymbol{w} = 0(\boldsymbol{u}+1)0$ with $\boldsymbol{u} \in \mathcal{C}_{n-2}(\pi)$, or
- $\boldsymbol{w} = 01^{n-k-2}0(\boldsymbol{u}+1)$ with $\boldsymbol{u} \in \mathcal{C}_k(\pi)$ for some k, $1 \le k \le n-3$.

In both of the first two cases, the number of words $\boldsymbol{w}$ is $c_{n-1}(\pi)$ and in the third case, this number is $c_{n-2}(\pi)$. In the last case, the number of words $\boldsymbol{w}$ is $\sum_{k=1}^{n-3} c_k(\pi)$. So, $c_n(\pi) = 2c_{n-1}(\pi) + c_{n-2}(\pi) + \sum_{k=1}^{n-3} c_k(\pi) = c_{n-1}(\pi) + \sum_{k=1}^{n-1} c_k(\pi)$, and with the initial conditions we have $c_n(\pi) = F_{2n-2}$. □

The sequence of Pell numbers $(p_n)_{n\ge 0}$ is defined as $p_0 = 0$, $p_1 = 1$ and $p_n = 2p_{n-1} + p_{n-2}$ for $n \ge 2$.

Proposition 17. *If $\pi = \{100, 101\}$, then $c_n(\pi)$ is the n*th *Pell number for $n \ge 1$ (A000129 in [2]).*

Proof. If $\boldsymbol{w} \in \mathcal{C}_n(\pi)$, $n \ge 2$, then either

- $\boldsymbol{w} = 0\boldsymbol{u}$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or
- $\boldsymbol{w} = 0(\boldsymbol{u}+1)$ with $\boldsymbol{u} \in \mathcal{C}_{n-1}(\pi)$, or
- $\boldsymbol{w} = 0(\boldsymbol{u}+1)0$ with $\boldsymbol{u} \in \mathcal{C}_{n-2}(\pi)$.

In both of the first two cases, the number of words $\boldsymbol{w}$ is $c_{n-1}(\pi)$ and it is $c_{n-2}(\pi)$ in the last case. So, $c_n(\pi)$ satisfies Pell numbers recurrence and considering its initial values the statement holds. □

3.5 Counting via generating function

Here we give bivariate generating functions $C_\pi(x, y)$ where the coefficient of $x^n y^k$ is the number of Catalan words of length n having k descents and avoiding π, for each of the remaining pairs π of patterns of length 3. Plugging $y = 1$ in $C_\pi(x, y)$ we obtain $C_\pi(x) = C_\pi(x, 1)$ where the coefficient of x^n is the number of Catalan words of length n avoiding π. All the obtained enumerating sequences are not yet recorded in [2], except that for: $\pi = \{100, 120\}$ and for $\pi = \{110, 120\}$ (see Corollary 3) and presumably for $\pi = \{100, 210\}$ (see Corollary 15). In almost all the proofs of the next propositions the desired generating function is the solution of a functional equation satisfied by it.

Proposition 18. *If $\pi = \{000, 021\}$, then*

$$C_\pi(x, y) = -\frac{x^4 y + x^2 y + 1}{x^2 + x - 1}.$$

Proof. Here we need the generating function for the Fibonacci numbers $C_{000,010}(x) = \frac{1}{1-x-x^2}$ for the set in Proposition 14. Note that words in $\mathcal{C}(000, 010)$ have no descents.

Let $\boldsymbol{w}$ be a non-empty word in $\mathcal{C}(\pi)$. Then $\boldsymbol{w}$ has one of the following forms:

- $\boldsymbol{w} = 0(\boldsymbol{u}+1)$ where $\boldsymbol{u} \in \mathcal{C}(000, 010)$; the generating function for these words is $x \cdot \frac{1}{1-x-x^2}$,
- $\boldsymbol{w} = 00(\boldsymbol{u}+1)$ where $\boldsymbol{u} \in C(000, 010)$; the generating function for these words is $x^2 \cdot \frac{1}{1-x-x^2}$,
- $\boldsymbol{w} = 0(\boldsymbol{u}+1)0$ where $\boldsymbol{u} \in \mathcal{C}(000, 010)$; the generating function for these words is $y \cdot x^2 \cdot \frac{1}{1-x-x^2}$,

– $\boldsymbol{w} = 0101(\boldsymbol{u}+2)$ where $\boldsymbol{u} \in \mathcal{C}(000, 010)$; the generating function for these words is $y \cdot x^4 \cdot \frac{1}{1-x-x^2}$.

Combining these cases and adding 1 corresponding to the empty word we have

$$C_\pi(x,y) = 1 + x(1+x) \cdot \frac{1}{1-x-x^2} + yx^2(1+x^2)\left(\frac{1}{1-x-x^2}\right),$$

which after calculation gives the desired result. □

Corollary 2. *If* $\pi = \{000, 021\}$*, then*

$$\begin{aligned} C_\pi(x) &= -\frac{x^4+x^2+1}{x^2+x-1} \\ &= 1 + x + 3x^2 + 4x^3 + 8x^4 + 12x^5 + 20x^6 + 32x^7 + O(x^6). \end{aligned}$$

Proposition 19. *If* $\pi = \{100, 120\}$*, then*

$$C_\pi(x,y) = -\frac{(x-1)^2}{x^3y - 2x^2 + 3x - 1}.$$

Proof. A word $\boldsymbol{w} \in \mathcal{C}(\pi)$ is in one of the following cases:

– $\boldsymbol{w}$ is a w.i. Catalan word,
– $\boldsymbol{w} = \boldsymbol{u}(m-1)(\boldsymbol{v}+m)$ where $\boldsymbol{u}$ is a w.i. Catalan word other than $00\cdots0$, m is the largest (last) entry of $\boldsymbol{u}$ and $\boldsymbol{v} \in C(\pi)$.

The generating function for the words of the first form is $\frac{1-x}{1-2x}$ and the generating function for the words of the second form is

$$\left(\frac{1-x}{1-2x} - \frac{1}{1-x}\right) \cdot x \cdot y \cdot C_\pi(x,y).$$

Combining these cases we deduce the functional equation below which solution gives the desired result:

$$C_\pi(x,y) = \left(\frac{1-x}{1-2x} - \frac{1}{1-x}\right) \cdot x \cdot y \cdot C_\pi(x,y) + \frac{1-x}{1-2x}.$$

□

Proposition 20. *If* $\pi = \{110, 120\}$*, then*

$$C_\pi(x,y) = -\frac{(x-1)^2}{x^3y - 2x^2 + 3x - 1}.$$

Proof. A word $\boldsymbol{w} \in \mathcal{C}(\pi)$ is in one of the following cases:

– $\boldsymbol{w}$ is a w.i. Catalan word,
– $\boldsymbol{w} = \boldsymbol{u}(m+1)\boldsymbol{v}$ where $\boldsymbol{u}$ is a non-empty w.i. Catalan word, m is the largest (last) entry of $\boldsymbol{u}$ and $\boldsymbol{v}$ is a word of the form $mm\cdots m(\boldsymbol{x}+m+1)$ with at least one m in its prefix and $\boldsymbol{x} \in \mathcal{C}(\pi)$.

The generating function for the words of the first form is $\frac{1-x}{1-2x}$.
For the second form, the generating function for the words $\boldsymbol{u}$ is $\frac{1-x}{1-2x} - 1 = \frac{x}{1-2x}$, and the generating function for the words $mm\cdots m(\boldsymbol{x}+m+1)$ is $\frac{x}{1-x}\cdot C_\pi(x,y)$. Thus, the generating function for the words of the second form is

$$\frac{x}{1-2x}\cdot x\cdot y\cdot \frac{x}{1-x}\cdot \mathcal{C}_\pi(x,y).$$

Combining these cases we deduce the functional equation

$$C_\pi(x,y) = \frac{x}{1-2x}\cdot x\cdot y\cdot \frac{x}{1-x}\cdot C_\pi(x,y) + \frac{1-x}{1-2x}.$$

□

The functional equations in the proofs of Propositions 19 and 20 are different but the resulting bivariate generating functions are the same. Instantiating y by 1 in $C_\pi(x,y)$ of these propositions we have the next corollary.

Corollary 3. *If $\pi = \{100, 120\}$ or $\pi = \{110, 120\}$, then*

$$\begin{aligned} C_\pi(x) &= -\frac{(x-1)^2}{x^3-2x^2+3x-1} \\ &= 1 + x + 2x^2 + 5x^3 + 12x^4 + 28x^5 + 65x^6 + 151x^7 + O(x^8), \end{aligned}$$

and $c_n(\pi)$ is the sequence A034943 *in [2].*

Proposition 21. *If $\pi = \{021, 110\}$, then*

$$C_\pi(x,y) = -\frac{x^5y + x^4y - x^4 - x^3y + 4x^3 - 6x^2 + 4x - 1}{(2x-1)(x-1)^3}.$$

Proof. A non-empty word $\boldsymbol{w} \in \mathcal{C}(\pi)$ is in one of the following cases:

- $\boldsymbol{w} = 0\boldsymbol{u}$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $x\cdot C_\pi(x,y)$,
- $\boldsymbol{w} = 0(\boldsymbol{u}+1)$ where $\boldsymbol{u}$ is a non-empty w.i. Catalan word; the generating function for these words is $x\cdot\frac{x}{1-2x}$,
- $\boldsymbol{w} = 01\boldsymbol{u}$ where $\boldsymbol{u}$ is a non-empty w.i. Catalan word; the generating function for these words is $x^2\cdot y\cdot\frac{x}{1-2x}$,
- $\boldsymbol{w} = \boldsymbol{u}0\cdots 0$ where $\boldsymbol{u}$ is a s.i. Catalan word of length at least three and $\boldsymbol{w}$ ending by at least one 0; the generating function for these words is $y\cdot\frac{x^4}{(1-x)^2}$.

Combining these cases and considering the empty word which contributes with 1 to $C_\pi(x,y)$, we deduce the functional equation

$$C_\pi(x,y) = 1 + x\cdot C_\pi(x,y) + x\cdot\frac{x}{1-2x} + x^2\cdot y\cdot\frac{x}{1-2x} + y\cdot\frac{x^4}{(1-x)^2}.$$

□

Corollary 4. *If* $\pi = \{021, 110\}$*, then*

$$\begin{aligned} C_\pi(x) &= -\frac{x^5 + 3x^3 - 6x^2 + 4x - 1}{(2x-1)(x-1)^3} \\ &= 1 + x + 2x^2 + 5x^3 + 12x^4 + 26x^5 + 53x^6 + 105x^7 + O(x^8), \end{aligned}$$

Proposition 22. *If* $\pi = \{110, 201\}$*, then*

$$C_\pi(x, y) = \frac{x^4y - x^3 + 3x^2 - 3x + 1}{(x-1)(x^3y - 2x^2 + 3x - 1)}.$$

Proof. A non-empty word $\boldsymbol{w} \in \mathcal{C}(\pi)$ is in one of the following cases:

- $\boldsymbol{w} = 0\boldsymbol{u}$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $x \cdot C_\pi(x, y)$,
- $\boldsymbol{w} = 0(\boldsymbol{u} + 1)$ where $\boldsymbol{u}$ is a non-empty word in $\mathcal{C}(\pi)$; the generating function for these words is $x \cdot (C_\pi(x, y) - 1)$,
- $\boldsymbol{w} = \boldsymbol{u}0 \cdots 0$ where $\boldsymbol{u}$ is a s.i. Catalan word of length at least 2 and $\boldsymbol{w}$ ending by at least one 0; the generating function for these words is $y \cdot \frac{x^3}{(1-x)^2}$,
- $\boldsymbol{w} = 010 \cdots 0(\boldsymbol{u} + 1)$ where $\boldsymbol{u}$ is a non-empty word in $\mathcal{C}(\pi)$ and $\boldsymbol{w}$ beginning by 010; the generating function for these words is $y \cdot x^3 \cdot \frac{1}{1-x} \cdot (C_\pi(x) - 1)$.

Combining these cases and adding 1 corresponding to the empty word we deduce the functional equation

$$C_\pi(x, y) = 1 + x \cdot C_\pi(x, y) + x \cdot (C_\pi(x, y) - 1) + y \cdot \frac{x^3}{(1-x)^2} + y \cdot x^3 \cdot \frac{1}{1-x} \cdot (C_\pi(x, y) - 1).$$

□

Corollary 5. *If* $\pi = \{110, 201\}$*, then*

$$\begin{aligned} C_\pi(x) &= \frac{x^4 - x^3 + 3x^2 - 3x + 1}{(x-1)(x^3 - 2x^2 + 3x - 1)} \\ &= 1 + x + 2x^2 + 5x^3 + 13x^4 + 32x^5 + 76x^6 + 178x^7 + O(x^8), \end{aligned}$$

Proposition 23. *If* $\pi = \{102, 201\}$*, then*

$$\mathcal{C}_\pi(x, y) = \frac{x^5y + x^4y^2 - x^5 - 5x^4y + 5x^4 + 6x^3y - 10x^3 - 2x^2y + 10x^2 - 5x + 1}{(-x+1)(x^2y - x^2 + 2x - 1)(x^2y - 2x^2 + 3x - 1)}.$$

Proof. A non-empty word $\boldsymbol{w} \in \mathcal{C}(\pi)$ is in one of the following cases:

- $\boldsymbol{w} = 0\boldsymbol{u}$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $x \cdot C_\pi(x, y)$,
- $\boldsymbol{w} = 0(\boldsymbol{u} + 1)$ where $\boldsymbol{u}$ is a non-empty word in $\mathcal{C}(\pi)$; the generating function for these words is $x \cdot (C_\pi(x, y) - 1)$,
- $\boldsymbol{w} = 0(\boldsymbol{v} + \mathbf{1})0 \cdots 0$ where $\boldsymbol{u}$ is a non-empty word in $\mathcal{C}(\pi)$ and $\boldsymbol{w}$ ending by at least one 0; the generating function for these words is $y \cdot x^2 \cdot \frac{1}{1-x} \cdot (C_\pi(x, y) - 1)$,

– $\boldsymbol{w} = 01\cdots1\boldsymbol{u}$ where $\boldsymbol{u}$ is a binary word beginning by a 0 and different from $0\cdots0$, or equivalently, $\boldsymbol{u}$ a word in $\mathcal{C}(012)$ other than $0\cdots0$; the generating function for these words is $\frac{x^2}{1-x}\cdot y\cdot\left(C_{012}(x,y)-\frac{1}{1-x}\right)=\frac{x^2\cdot y}{1-x}\cdot\left(\frac{1-x+x^2-x^2y}{1-2x+x^2-x^2y}-\frac{1}{1-x}\right)$, see Theorem 4 in [24].

Combining these cases and adding 1 corresponding to the empty word we deduce the functional equation

$$\begin{aligned}C_\pi(x,y)= \quad & 1+x\cdot C_\pi(x,y)+x\cdot(C_\pi(x,y)-1)+\frac{x^2\cdot y}{1-x}\cdot(C_\pi(x,y)-1)\\ &+\frac{x^2\cdot y}{1-x}\cdot\left(\frac{1-x+x^2-x^2y}{1-2x+x^2-x^2y}-\frac{1}{1-x}\right).\end{aligned}$$

□

Corollary 6. *If* $\pi=\{102,201\}$*, then*

$$\begin{aligned}C_\pi(x)&=\frac{x^4-4x^3+8x^2-5x+1}{(x-1)(2x-1)(x^2-3x+1)}\\&=1+x+2x^2+5x^3+14x^4+40x^5+113x^6+314x^7+O(x^8),\end{aligned}$$

Proposition 24. *If* $\pi=\{100,110\}$*, then*

$$C_\pi(x,y)=\frac{x^4y-x^4+2x^3-2x+1}{(x-1)(x^3y-2x^3+x^2+2x-1)}.$$

Proof. A non-empty word in $\mathcal{C}(\pi)$ has one of the following forms:

– $0\boldsymbol{u}$ where $\boldsymbol{u}\in\mathcal{C}(\pi)$; the generating function for these words is $x\cdot C_\pi(x,y)$,
– $0(\boldsymbol{u}+1)$ where $\boldsymbol{u}$ is a non-empty word in $\mathcal{C}(\pi)$; the generating function for these words is $x\cdot(C_\pi(x,y)-1)$,
– $\boldsymbol{u}(m+1)(m+2)\boldsymbol{v}$ where $\boldsymbol{u}$ and $\boldsymbol{v}$ are non-empty s.i. Catalan words, m is the largest entry of $\boldsymbol{u}$ and the length of $\boldsymbol{v}$ is less than or equal to that of $\boldsymbol{u}$; the generating function for these words is $y\cdot x^4\cdot(x+1)\cdot\frac{1}{(1-x^2)^2}$,
– $\boldsymbol{u}(m+1)\boldsymbol{u}(\boldsymbol{v}+m+1)$ where $\boldsymbol{u}$ is a non-empty s.i. Catalan word, m is the largest entry of $\boldsymbol{u}$ and $\boldsymbol{v}\in\mathcal{C}(\pi)$; the generating function for these words is $y\cdot x^3\cdot\frac{1}{1-x^2}\cdot C_\pi(x,y)$.

Combining these cases and adding 1 corresponding to the empty word we deduce the functional equation

$$C_\pi(x,y)=1+x\cdot C_\pi(x,y)+x\cdot(C_\pi(x,y)-1)+y\cdot x^4\cdot\frac{(x+1)}{(1-x^2)^2}+y\cdot x^3\cdot\frac{1}{1-x^2}\cdot C_\pi(x,y).$$

□

Corollary 7. *If* $\pi=\{100,110\}$*, then*

$$\begin{aligned}C_\pi(x)&=\frac{-2x^3+2x-1}{(x-1)(x^3-x^2-2x+1)}\\&=1+x+2x^2+5x^3+12x^4+28x^5+64x^6+145x^7+O(x^8).\end{aligned}$$

Proposition 25. *If* $\pi = \{000, 110\}$*, then*

$$C_\pi(x,y) = \frac{x^3y + x^3 - x^2 - x + 1}{(-x+1)(x^3 - x^2y - x^2 - x + 1)}.$$

Proof. A non-empty word in $\mathcal{C}(\pi)$ has one of the following forms:

- $0(\boldsymbol{u}+1)$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $x \cdot C_\pi(x,y)$,
- $\boldsymbol{u}\boldsymbol{u}(\boldsymbol{v}+m+1)$ where $\boldsymbol{u}$ is a non-empty s.i. Catalan word, m is the largest (last) entry of $\boldsymbol{u}$ and $\boldsymbol{v} \in \mathcal{C}(\pi)$; the generating function for these words is $y \cdot \frac{x^2}{1-x^2} \cdot C_\pi(x,y)$,
- $\boldsymbol{u}(m+1)\boldsymbol{v}$ where $\boldsymbol{u}$ and m are as above, and $\boldsymbol{v}$ is a non-empty s.i. Catalan word of length less than that of $\boldsymbol{u}$; the generating function for these words is $y \cdot x^3 \cdot (1+x) \cdot \frac{1}{(1-x^2)^2}$.

Combining these cases and adding 1 corresponding to the empty word we deduce the functional equation

$$C_\pi(x,y) = 1 + x \cdot C_\pi(x,y) + y \cdot \frac{x^2}{1-x^2} \cdot C_\pi(x,y) + y \cdot x^3 \cdot (1+x) \cdot \frac{1}{(1-x^2)^2}.$$

□

Corollary 8. *If* $\pi = \{000, 110\}$*, then*

$$\begin{aligned} C_\pi(x) &= \frac{2x^3 - x^2 - x + 1}{(-x+1)(x^3 - 2x^2 - x + 1)} \\ &= 1 + x + 2x^2 + 4x^3 + 8x^4 + 15x^5 + 28x^6 + 51x^7 + O(x^8), \end{aligned}$$

Proposition 26. *If* $\pi = \{000, 102\}$ *or* $\pi = \{000, 201\}$*, then*

$$C_\pi(x,y) = \frac{yx^2 - 1}{yx^4 + yx^2 + x^2 + x - 1}.$$

Proof. If $\pi = \{000, 102\}$, then a non-empty word in $\mathcal{C}(\pi)$ has one of the following forms:

- $0(\boldsymbol{u}+1)$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $x \cdot C_\pi(x,y)$,
- $00(\boldsymbol{u}+1)$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $x^2 \cdot C_\pi(x,y)$,
- $0(\boldsymbol{u}+1)0$ where $\boldsymbol{u}$ is a non-empty word in $\mathcal{C}(\pi)$; the generating function for these words is $y \cdot x^2 \cdot (C_\pi(x,y) - 1)$,
- $01(\boldsymbol{u}+2)01$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $y \cdot x^4 \cdot C_\pi(x,y)$.

Similarly, if $\pi = \{000, 201\}$ and $\boldsymbol{w}$ is a non-empty word in $\mathcal{C}(\pi)$, then $\boldsymbol{w}$ has either one of the first three forms above, or

- $\boldsymbol{w} = 0101(\boldsymbol{u}+2)$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $y \cdot x^4 \cdot C_\pi(x,y)$.

In both cases we obtain the functional equation

$$C_\pi(x,y) = 1 + x \cdot C_\pi(x,y) + x^2 \cdot C_\pi(x,y) + y \cdot x^2 \cdot (C_\pi(x,y) - 1) + y \cdot x^4 \cdot C_\pi(x,y).$$

□

Corollary 9. *If* $\pi = \{000, 102\}$ *or* $\pi = \{000, 201\}$*, then*

$$\begin{aligned} C_\pi(x) &= \frac{x^2 - 1}{x^4 + 2x^2 + x - 1} \\ &= 1 + x + 2x^2 + 4x^3 + 9x^4 + 18x^5 + +38x^6 + 78x^7 + O(x^8), \end{aligned}$$

Proposition 27. *If* $\pi = \{000, 120\}$*, then*

$$C_\pi(x, y) = -\frac{x^4y + x^3y + 1}{x^4y + x^2 + x - 1}.$$

Proof. A non-empty word in $\mathcal{C}(\pi)$ has one of the following forms:

- $0(\boldsymbol{u} + 1)$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $x \cdot C_\pi(x, y)$,
- $00(\boldsymbol{u} + 1)$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $x^2 \cdot C_\pi(x, y)$,
- $0101(\boldsymbol{u} + 2)$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $y \cdot x^4 \cdot C_\pi(x, y)$.

Apart from these general cases, there are two other fixed length ones:

- 010; the corresponding generating function is $y \cdot x^3$,
- 0110; the corresponding generating function is $y \cdot x^4$.

Combining these cases and adding 1 corresponding to the empty word we deduce the functional equation

$$C_\pi(x, y) = 1 + x \cdot C_\pi(x, y) + x^2 \cdot C_\pi(x, y) + y \cdot x^4 \cdot C_\pi(x, y) + y \cdot x^3 + y \cdot x^4.$$

□

Corollary 10. *If* $\pi = \{000, 120\}$*, then*

$$\begin{aligned} C_\pi(x) &= -\frac{x^4 + x^3 + 1}{x^4 + x^2 + x - 1} \\ &= 1 + x + 2x^2 + 4x^3 + 8x^4 + 13x^5 + 23x^6 + 40x^7 + O(x^8), \end{aligned}$$

Proposition 28. *If* $\pi = \{201, 210\}$*, then*

$$C_\pi(x, y) = \frac{x^4y - 2x^4 - 3x^3y + 7x^3 + x^2y - 9x^2 + 5x - 1}{(2x - 1)(x - 1)(x^2y - 2x^2 + 3x - 1)}.$$

Proof. A non-empty word $\boldsymbol{w}$ in $\mathcal{C}(\pi)$ has one of the following forms:

- $0\boldsymbol{u}$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $x \cdot C_\pi(x, y)$,
- $0(\boldsymbol{u} + 1)$ where $\boldsymbol{u}$ is a non-empty word in $\mathcal{C}(\pi)$; the generating function for these words is $x \cdot (C_\pi(x, y) - 1)$,
- $01 \cdots 1\boldsymbol{u}$ where $\boldsymbol{u}$ is a non-empty word in $\mathcal{C}(\pi)$ and 01 is a prefix of $\boldsymbol{w}$; the generating function for these words is $y \cdot \frac{x^2}{1-x} \cdot (C_\pi(x, y) - 1)$,
- $0(\boldsymbol{u} + 1)0 \ldots 0$ where $\boldsymbol{u}$ is a w.i. Catalan word other than $0 \cdots 0$ and $\boldsymbol{w}$ ending by a 0; the generating function for these words is $y \cdot \frac{x^2}{1-x} \cdot (\frac{1-x}{1-2x} - \frac{1}{1-x})$.

Combining these cases and adding 1 corresponding to the empty word we deduce the functional equation

$$C_\pi(x,y) = \quad 1 + x\cdot C_\pi(x,y) + x\cdot(C_\pi(x,y)-1) + y\cdot\frac{x^2}{1-x}\cdot(C_\pi(x,y)-1) + y\cdot\frac{x^2}{1-x}\cdot\left(\frac{1-x}{1-2x}-\frac{1}{1-x}\right).$$

□

Corollary 11. *If* $\pi = \{201, 210\}$*, then*

$$\begin{aligned} C_\pi(x) &= \frac{x^4-4x^3+8x^2-5x+1}{(x-1)(2x-1)(x^2-3x+1)} \\ &= 1+x+2x^2+5x^3+14x^4+40x^5+113x^6+314x^7+O(x^8), \end{aligned}$$

Proposition 29. *If* $\pi = \{102, 210\}$*, then*

$$C_\pi(x,y) = \frac{(2y-2)x^7+(13-10y-y^2)x^6-36x^5+19x^5y+55x^4-17x^4y-50x^3+7x^3y+27x^2-yx^2-8x+1}{(x-1)^3(2x-1)^2(x^2y-x^2+2x-1)}.$$

Proof. A non-empty word $\boldsymbol{w}$ in $\mathcal{C}(\pi)$ has one of the following forms:

- $0\boldsymbol{u}$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $x\cdot C_\pi(x,y)$,
- $0(\boldsymbol{u}+1)$ where $\boldsymbol{u}$ is a non-empty word in $\mathcal{C}(\pi)$; the generating function for these words is $x\cdot(C_\pi(x,y)-1)$,
- $01\cdots1\boldsymbol{u}$ where $\boldsymbol{u}$ is a non-empty word in $\mathcal{C}(012)$, and $\boldsymbol{w}$ begins by 01; the generating function for these words is $y\cdot\frac{x^2}{1-x}\cdot(C_{012}(x,y)-1) = y\cdot\frac{x^2}{1-x}\cdot(\frac{1-x+x^2-x^2y}{1-2x+x^2-x^2y}-1)$ (see Theorem 4 in [24] for the generating function of $C_{012}(x,y)$),
- $0(\boldsymbol{u}+1)\boldsymbol{v}$ where $\boldsymbol{u}$ is a w.i. Catalan word of length at least 2 different from $0\cdots0$ and $\boldsymbol{v}$ is a non-empty word in $\mathcal{C}(010,012)$ (see Proposition 5); the generating function for these words is $x\cdot(\frac{1-x}{1-2x}-\frac{1}{1-x})\cdot y\cdot C_{010,012}(x,y) = y\cdot x\cdot\frac{x}{(1-x)^2}\cdot(\frac{1-x}{1-2x}-\frac{1}{1-x})$.

Combining these cases and adding 1 corresponding to the empty word we deduce the functional equation

$$C_\pi(x,y) = \quad 1 + x\cdot C_\pi(x,y) + x\cdot(C_\pi(x,y)-1) + y\cdot\frac{x^2}{1-x}\cdot\left(\frac{1-x+x^2-x^2y}{1-2x+x^2-x^2y}-1\right) + y\cdot\frac{x^2}{(1-x)^2}\cdot\left(\frac{1-x}{1-2x}-\frac{1}{1-x}\right).$$

□

Corollary 12. *If $\pi = \{102, 210\}$, then*

$$\begin{aligned} C_\pi(x) &= \frac{2x^6 - 17x^5 + 38x^4 - 43x^3 + 26x^2 - 8x + 1}{(x-1)^3(2x-1)^3} \\ &= 1 + x + 2x^2 + 5x^3 + 14x^4 + 40x^5 + 111x^6 + 295x^7 + O(x^8). \end{aligned}$$

Proposition 30. *If $\pi = \{100, 102\}$, then*

$$C_\pi(x, y) = \frac{x^5y - x^4y - x^3 + 2x^3y + 3x^2 - x^2y - 3x + 1}{(x-1)(x^4y - x^3y - 2x^2 + x^2y + 3x - 1)}.$$

Proof. A non-empty word in $\mathcal{C}(\pi)$ has one of the following forms:

- $0\boldsymbol{u}$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $x \cdot C_\pi(x, y)$,
- $0(\boldsymbol{u} + 1)$ where $\boldsymbol{u}$ is a non-empty word in $\mathcal{C}(\pi)$; the generating function for these words is $x \cdot (C_\pi(x, y) - 1)$,
- $0(\boldsymbol{u}+1)0$ where $\boldsymbol{u}$ is as above; the generating function for these words is $y \cdot x^2 \cdot (C_\pi(x, y) - 1)$,
- $011 \cdots 1(\boldsymbol{u} + 2)01$ where $\boldsymbol{u}$ is as above; the generating function for these words is $y \cdot \frac{x^4}{1-x}(C_\pi(x, y) - 1)$,
- $\boldsymbol{uv}$ where $\boldsymbol{u}$ and $\boldsymbol{v}$ are binary words of length at least 2 of form $011 \cdots 1$; the generating function for these words is $y \cdot \frac{x^4}{(1-x)^2}$.

Combining these cases and adding 1 corresponding to the empty word we deduce the functional equation

$$\begin{aligned} C_\pi(x, y) = \quad & 1 + x \cdot C_\pi(x, y) + x \cdot (C_\pi(x, y) - 1) + y \cdot x^2 \cdot (C_\pi(x, y) - 1) + \\ & y \cdot \frac{x^4}{1-x}(C_\pi(x, y) - 1) + y \cdot \frac{x^4}{(1-x)^2} \end{aligned}$$

□

Corollary 13. *If $\pi = \{100, 102\}$, then*

$$\begin{aligned} C_\pi(x) &= \frac{x^5 - x^4 + x^3 + 2x^2 - 3x + 1}{(x-1)(x^4 - x^3 - x^2 + 3x - 1)} \\ &= 1 + x + 2x^2 + 5x^3 + 13x^4 + 34x^5 + 87x^6 + 220x^7 + O(x^8), \end{aligned}$$

In the proof of the next proposition, we need the following lemma where the generating functions for some particular subsets of $\mathcal{C}(000, 210)$ are given.

Lemma 2. *The bivariate generating function corresponding to*

1. *the set $\mathcal{A}$ of words $\boldsymbol{uu}$ with $\boldsymbol{u}$ a non-empty s.i. Catalan word is $A(x, y) = x^2 + yx^2 \cdot \frac{x^2}{1-x^2}$ and $A(x) = \frac{x^2}{1-x^2}$;*
2. *the set $\mathcal{B}$ of words $\boldsymbol{uv}$ with $\boldsymbol{u}$ and $\boldsymbol{v}$ non-empty s.i. Catalan words and the length of $\boldsymbol{v}$ is less than or equal to that of $\boldsymbol{u}$ is $B(x, y) = y \cdot \frac{x^2}{1-x^2} \cdot \frac{1}{1-x}$;*
3. *the set $\mathcal{D}$ of words $\boldsymbol{uv}$ with $\boldsymbol{u}$ and $\boldsymbol{v}$ non-empty s.i. Catalan words and the length of $\boldsymbol{v}$ is less than that of $\boldsymbol{u}$ is $D(x, y) = y \cdot \frac{x^2}{1-x^2} \cdot \frac{x}{1-x}$.*

Proof. 1. For any even n there is exactly one word of this form, so the monovariate corresponding generating function is $\frac{x^2}{1-x^2}$; and only words of length larger than two have one descent.

2. The transformation $(\boldsymbol{uu}, \boldsymbol{x}) \mapsto \boldsymbol{uxu}$ where $\boldsymbol{uu} \in \mathcal{A}$ and $\boldsymbol{x}$ is a s.i. Catalan word defines a bijection between pairs of such words and $\mathcal{B}$, and thus $B(x, y) = y \cdot A(x) \cdot \frac{1}{1-x}$.

3. Similarly as point 2. □

Proposition 31. *If* $\pi = \{000, 210\}$*, then*

$$\mathcal{C}_\pi(x, y) = -\frac{(x+1)(x^3 + x^3y - 2x + 1)}{(x^4 - x^4y + x^3 - 2x^2 - x + 1)(x^2 + x - 1)}.$$

Proof. A non-empty word in $\mathcal{C}(000, 210)$ has one of the following forms:

- $\boldsymbol{u}$ with $\boldsymbol{u} \in \mathcal{D}$ and $\mathcal{D}$ as in Lemma 2; the generating function for these words is $D(x, y) = y \cdot \frac{x^2}{1-x^2} \cdot \frac{x}{1-x}$,
- $\boldsymbol{u}(m+1)(m+1)(\boldsymbol{x} + m + 2)\boldsymbol{v}$ with $\boldsymbol{u}$ and $\boldsymbol{v}$ non-empty s.i. Catalan words and the length of $\boldsymbol{v}$ is less than or equal to that of $\boldsymbol{u}$, m is the largest entry of $\boldsymbol{u}$, and $\boldsymbol{x} \in \mathcal{C}(000, 010)$; the generating function for these words is $B(x, y) \cdot x^2 \cdot \frac{1}{1-x-x^2} = y \cdot \frac{x^2}{1-x^2} \cdot \frac{1}{1-x} \cdot x^2 \cdot \frac{1}{1-x-x^2}$ (see Lemma 2 and Proposition 14),
- $0(\boldsymbol{u} + 1)$ where $\boldsymbol{u} \in \mathcal{C}(\pi)$; the generating function for these words is $x \cdot C_\pi(x, y)$,
- $\boldsymbol{uu}(\boldsymbol{v} + m + 1)$ where $\boldsymbol{u}$ is a non-empty s.i. Catalan word, m the largest entry of $\boldsymbol{u}$ and $\boldsymbol{v} \in \mathcal{C}(\pi)$; the generating function for these words is $A(x, y) \cdot C_\pi(x, y)$.

Combining these cases and adding 1 corresponding to the empty word we deduce the functional equation

$$C_\pi(x, y) = 1 + y \cdot \frac{x^2}{1-x^2} \cdot \frac{x}{1-x} + y \cdot \frac{x^4}{1-x^2} \cdot \frac{1}{1-x} \cdot \frac{1}{1-x-x^2} + x \cdot C(x, y) + \left(x^2 + y \cdot \frac{x^4}{1-x^2}\right) \cdot C(x, y)$$

□

Corollary 14. *If* $\pi = \{000, 210\}$*, then*

$$\begin{aligned} C_\pi(x) &= -\frac{(x+1)(2x^3 - 2x + 1)}{(x^3 - 2x^2 - x + 1)(x^2 + x - 1)} \\ &= 1 + x + 2x^2 + 4x^3 + 9x^4 + 18x^5 + 37x^6 + 72x^7 + O(x^8), \end{aligned}$$

In the proof of the next proposition we need the following lemma where the generating functions for two subsets of $\mathcal{C}(100, 210)$ are given.

Lemma 3.

1. *The generating function corresponding to the set $\mathcal{E}$ of words $\boldsymbol{uv}$ with $\boldsymbol{u}$ a w.i. Catalan word, $\boldsymbol{v}$ a non-empty s.i. Catalan word and the largest entry of $\boldsymbol{v}$ is equal to that of $\boldsymbol{u}$ minus 1 is* $\frac{x^3}{(x-1)(x^2+x-1)}$.
2. *The generating function corresponding to the set $\mathcal{F}$ of words $\boldsymbol{uv}$ with $\boldsymbol{u}$ a w.i. Catalan word, $\boldsymbol{v}$ a non-empty s.i. Catalan word and the largest entry of $\boldsymbol{v}$ is less than that of $\boldsymbol{u}$ minus 1 is* $\frac{x^3}{(x-1)(x^2+x-1)} \cdot \frac{x}{1-2x}$.

Proof. 1. If $\mathcal{E}_n$ is the set of words of length n in $\mathcal{E}$, then $\mathcal{E}_n = \varnothing$ for $0 \leq i \leq 2$, $\mathcal{E}_3 = \{010\}$ and $\mathcal{E}_4 = \{0010, 0110\}$. With $\boldsymbol{u}$ and $\boldsymbol{v}$ as above, for any $n \geq 3$, the transformation

$\boldsymbol{uv} \mapsto \boldsymbol{u}a\boldsymbol{v}$, with a the maximal entry of $\boldsymbol{u}$, transforms a word in $\mathcal{E}_n$ into one in $\mathcal{E}_{n+1}$ where the maximal entry occurs at least twice,

$\boldsymbol{uv} \mapsto \boldsymbol{u}(a+1)\boldsymbol{v}(b+1)$, with a and b the maximal entries of $\boldsymbol{u}$ and of $\boldsymbol{v}$ respectively, transforms a word in $\mathcal{E}_n$ into one in $\mathcal{E}_{n+2}$ where the maximal entry occurs once.

Any word in $\mathcal{E}_n$, $n \geq 5$, except $0\cdots010$, can be obtained uniquely from either a word in $\mathcal{E}_{n-1}$ or in $\mathcal{E}_{n-2}$ by one of these transformations. This yields the recurrence $|\mathcal{E}_n| = 1 + |\mathcal{E}_{n-1}| + |\mathcal{E}_{n-2}|$ for $n \geq 5$, and the desired generating function is precisely that of the sequence $(|\mathcal{E}_n|)_{n\geq 0}$.

2. Any pair of words $(\boldsymbol{w}, \boldsymbol{x})$ with $\boldsymbol{w} = \boldsymbol{uv} \in \mathcal{E}$ (with $\boldsymbol{u}$ and $\boldsymbol{v}$ as above) and $\boldsymbol{x}$ a non-empty w.i. Catalan word can be transformed into the word $\boldsymbol{uxv} \in \mathcal{F}$, and $(\boldsymbol{w}, \boldsymbol{x}) \mapsto \boldsymbol{uxv}$ is a bijection, so the generating function for $\mathcal{F}$ is that for $\mathcal{E}$ multiplied by $\frac{x}{1-2x}$.

□

Proposition 32. *If $\pi = \{100, 210\}$, then*

$$C_\pi(x,y) = \frac{1-x}{1-2x} - \frac{x^3 y}{(2x-1)(2x^3 - x^3 y + x^2 - 3x + 1)}.$$

Proof. First, we consider only words in $\mathcal{C}(\pi)$ having at least one descent, and we denote by $G(x,y)$ the corresponding generating function, and clearly $C_\pi(x,y) = \frac{1-x}{1-2x} + G(x,y)$.

A word in $\mathcal{C}(\pi)$ with at least one descent has one of the following forms:

- $\boldsymbol{u}(\boldsymbol{\alpha}+s+1)(\boldsymbol{v}+s+t+1)$ where $\boldsymbol{u}$ and $\boldsymbol{v}$ are both w.i. Catalan words, $\boldsymbol{\alpha}$ belongs to the set $\mathcal{E}$ defined in Lemma 3, and s is the largest symbol of $\boldsymbol{u}$ (and for convenience -1 if $\boldsymbol{u}$ is empty) and t that of $\boldsymbol{\alpha}$; the generating function for these words is $y \cdot \frac{x^3}{(x-1)(x^2+x-1)} \cdot \left(\frac{1-x}{1-2x}\right)^2$,
- $\boldsymbol{u}(\boldsymbol{\alpha} + s + 1)$ where $\boldsymbol{u}$ and s are as above, and $\boldsymbol{\alpha}$ belongs to the set $\mathcal{F}$ defined in Lemma 3; the generating function for these words is $y \cdot \frac{1-x}{1-2x} \cdot \frac{x^3}{(x-1)(x^2+x-1)} \cdot \frac{x}{1-2x}$,
- $\boldsymbol{u}(\boldsymbol{\alpha} + s + 1)(\boldsymbol{v} + s + t + 1)$ where $\boldsymbol{u}$ and s are as above, $\boldsymbol{\alpha}$ belongs to $\mathcal{E}$, $\boldsymbol{v}$ is a word in $\mathcal{C}(\pi)$ with at least one descent, and t is the largest symbol of $\boldsymbol{\alpha}$; the generating function for these words is $y \cdot \frac{1-x}{1-2x} \cdot \frac{x^3}{(x-1)(x^2+x-1)} \cdot G(x,y)$.

It follows that $G(x,y)$ satisfies the functional equation

$$\begin{aligned} G(x,y) \;=\; & y \cdot \frac{x^3}{(x-1)(x^2+x-1)} \cdot \left(\frac{1-x}{1-2x}\right)^2 + y \cdot \frac{1-x}{1-2x} \cdot \frac{x^3}{(x-1)(x^2+x-1)} \cdot \frac{x}{1-2x} + \\ & y \cdot \frac{1-x}{1-2x} \cdot \frac{x^3}{(x-1)(x^2+x-1)} \cdot G(x,y). \end{aligned}$$

Finally, solving it and adding the generating function for the Catalan words with no descents (that is, w.i. Catalan words) the statement holds.

□

Corollary 15. *If* $\pi = \{100, 210\}$*, then*

$$\begin{aligned} C_\pi(x) &= \frac{x-1}{2x-1} - \frac{x^3}{(2x-1)(x-1)(x^2+2x-1)} \\ &= 1 + x + 2x^2 + 5x^3 + 13x^4 + 34x^5 + 88x^6 + 225x^7 + O(x^8), \end{aligned}$$

Numerical evidences let us believe that $c_n(100, 210)$ is the sequence A267905 in [2], however we failed to prove this formally.

3.6 Final remarks

Catalan words are in bijection with Dyck paths and thus pattern avoiding Catalan words correspond to restricted Dyck paths. For instance, a Catalan word avoiding 012 corresponds to a Dyck path of height at most two. In this context, it can be of interest to investigate how our results on pattern avoiding Catalan words translate to corresponding restricted Dyck paths.

Even if in this article we restrict ourselves to the avoidance of two patterns of length 3, some classes considered here can be trivially extended to larger length patterns, for instance $\mathcal{C}(102, 201) = \mathcal{C}(01012, 01201)$. In this light, it can be of interest to explore Catalan words avoiding patterns of length 4 or more, triples of patterns or generalized patterns.

3.7 Conclusion

In this contribution, we were interested in the enumeration of Catalan words avoiding pairs of patterns of length three. Catalan words are particular growth-restricted words counted by the eponymous integer sequence. Enumerating results and proofs of each case were presented with various enumeration methods, by constructive bijections or bivariate generating functions with respect to the length and descent number.

$\sigma\backslash\tau$	000	001	010	011	012	021	100	101	102	110	120	201	210
000	-	P. 14	P. 14	$u.c.$	$u.c.$	*C. 2*	s	P. 7	*C. 9*	*C. 8*	*C. 10*	*C. 9*	*C. 14*
001	-	-	P. 5	P. 5	P. 5	P. 13	P. 15	s	s	P. 13	P. 13	s	P. 12
010	-	-	-	P. 5	P. 5	s	s	s	s	s	s	s	s
011	-	-	-	-	P. 5	s	P. 6	s	s	s	P. 6	s	s
012	-	-	-	-	-	s	P. 13	P. 13	s	P. 13	s	s	s
021	-	-	-	-	-	-	P. 9	P. 9	P. 11	*C. 4*	P. 10	s	s
100	-	-	-	-	-	-	-	P. 17	*C. 13*	*C. 7*	C. 3	P. 16	C. 15
101	-	-	-	-	-	-	-	-	s	P. 9	P. 9	s	P. 8
102	-	-	-	-	-	-	-	-	-	P. 11	P. 8	*C. 6*	*C. 12*
110	-	-	-	-	-	-	-	-	-	-	C. 3	*C. 5*	s
120	-	-	-	-	-	-	-	-	-	-	-	s	s
201	-	-	-	-	-	-	-	-	-	-	-	-	*C. 11*
210	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 3.2: Pairs $\{\sigma, \tau\}$ where τ is superfluous for σ are marked by s and those yielding ultimately constant enumerating sequences (see Proposition 4) by $u.c.$. The references are to the propositions or the corollaries where the enumerating sequences or generating functions are given. Pairs referred by the same proposition or corollary form a Wilf-equivalence class and enumerating results that are not yet recorded in [2] are italicized. Highlighted pairs are already enumerated in [30] in the context of ascent sequences, see Section 3.1.

Chapter 4

Stack-sorting permutations with stacks under constraints

The study of sorting permutation (arranging permutations in increasing order) began with the third volume from Knuth's Art of Computer Programming [124], which analyzes a certain "stack sorting algorithm", then he introduced serial composition of stacks and gave many examples of links between the worst/best/average case behavior of sorting algorithms and the combinatorial structures hidden behind permutations [125]. We refer also to [38] and chapter 8 of [39] by Bóna for introductions to the area of sorting we are interested in.

There is a long line of papers in the computer science literature dedicated to problems of *sorting permutations* with different devices, e.g., *stacks*, *queues*, and *deques*, see for example [8, 12, 38, 41, 44, 53, 195].

After Knuth introduced the problem of stack-sorting in 1968 [124], stack-sorting was further generalized to sorting networks by Tarjan [208] while several variants appear by either considering other types of combinatorial structures or by changing rules [9, 95, 184]. Then West [221] defined a deterministic variant of Knuth's algorithm.

4.1 Sorting with t-stacks in series

A *stack* is a last-in first-out linear sorting device with *push / insert* and *pop / remove* operations. In other words, a stack is a container for a linear sequence (in this thesis, for a permutation) that one is allowed to modify by inserting new elements, one at a time, in its tail and removing tail elements, also , one at a time. Initially, the stack is empty and then a sequence of interlaced insertions with removals is performed. Thus an input permutation is transformed thereby into an output permutation.

In his dissertation, West [221] defined a deterministic variant of Knuth's algorithm. This variant is a function, which we call the "stack-sorting map" and denote by s, that sends permutations to permutations. To define the function s, let us begin with an input permutation $w = w_1 w_2 \dots w_n$. At any time during this procedure, if the next entry in the input permutation is smaller than the entry at the top of the stack or if the stack is empty, the next entry in the input permutation is placed to the top of the stack. Otherwise, the entry at the top of the stack is appended to the end of the

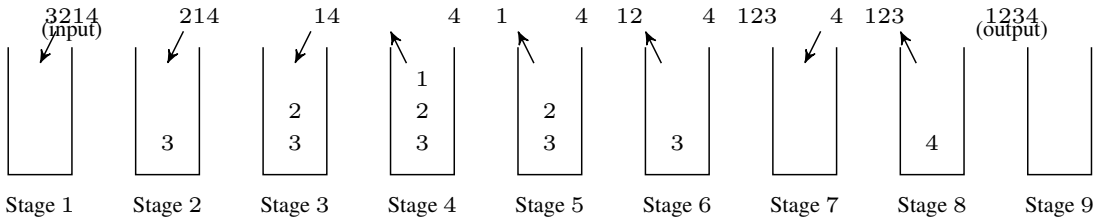


Figure 4.1: Stack sorting the permutation 3214

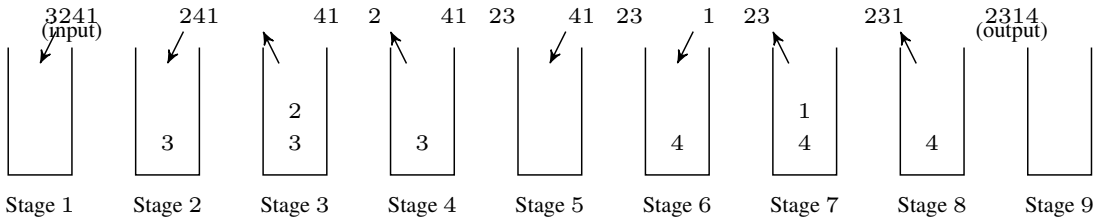


Figure 4.2: Stack sorting the permutation 3241

ascending output permutation. This process terminates when the output permutation has length n, and $s(w)$ is defined to be this output permutation.

Definition 13. We say a permutation w is t-stack-sortable if $s^t(w)$ is an increasing permutation, where s^t denotes the t-fold iterate of s. Let $W_t(n)$ be the set of t-stack-sortable permutations in $\mathfrak{S}_n$ and $\mathrm{W}_t(n) = |W_t(n)|$

We remind that $des(w)$ denote the number of descents of w.

Fig. 4.1 shows an example of a stack-sortable permutation, while Fig. 4.2 shows an example of a non-stack-sortable permutation. The permutation 3241 is a 3-stack sortable permutation, see Fig. 4.3

Knuth simultaneously initiated the study of stack-sorting and the investigation of permutation patterns with the following theorem.

Definition 14. [125] A permutation is 1-stack-sortable if and only if it avoids the pattern 231. Furthermore, $\mathrm{W}_1(n) = |\mathfrak{S}_n(231)| = c_n$, where c_n is the nth Catalan number $\frac{1}{n+1}\binom{2n}{n}$

In his celebrated book [124], Knuth gave the following characterization of stack sortable permutations, which is often considered the starting point of stack sorting and permutation patterns disciplines.

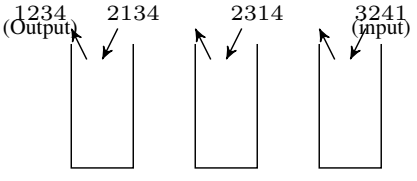


Figure 4.3: Sorting the permutation 3241 with 3 stacks in series

Proposition 1 ([124]). A permutation π is sortable using a classical stack (that is, a 21-avoiding stack) if and only if π avoids the pattern 231.

In his dissertation, West conjectured a formula for $W_2(n)$, which Zeilberger later proved.

Definition 15. [233] We have

$$\mathrm{W}_2(n) = \frac{2}{(n+1)(2n+1)}\binom{3n}{n}. \tag{4.1}$$

Combinatorial proofs of Zeilberger's theorem emerged later in [69, 89, 90, 108]. Some authors have investigated the enumeration of 2-stack-sortable permutations according to various statistics [37, 42, 45, 69].

For $t \geq 3$, there is very little known about t-stack-sortable permutations when $t \geq 3$ is fixed. Ulfarsson [215] characterized 3-stack-sortable permutations in terms of new "decorated patterns", but the characterization is too unwieldy to yield any additional information. The recent paper [10] shows that for every $t \geq 1$, the set of t-stack-sortable permutations can be described by a sentence in a first-order logical theory that the authors call TOTO. The paper [65] also investigates t-stack-sortable permutations when $t = n - r$ for some fixed r (focusing on the case in which $r = 4$). For fixed $t \geq 3$, the best known general upper bound for $\mathrm{W}_t(n)$, see Theorem 3.4 of [38], is the estimate

$$\mathrm{W}_t(n) \leq (t+1)^{2n}. \tag{4.2}$$

Bóna has also shown that

$$\lim_{n\longrightarrow\infty} \mathrm{W}_3(n)^{\frac{1}{n}} < 12.53296, \quad \textit{and} \quad \lim_{n\longrightarrow\infty} \mathrm{W}_4(n)^{\frac{1}{n}} < 21.97225 \tag{4.3}$$

Even though the problem of one stack is or may be simple, things got to be significantly more complicated in case one allows more stacks associated in series. Unnecessary to say, the enumeration of t-stack sortable permutations is obscure. Since the issue of sorting with two stacks is as well troublesome, a few special cases have been considered. Among them, the West-2-stack-sortable permutations [221] are those permutations that can be sorted by making two passes through a stack. Equivalently, these are the permutations that can be sorted by 2 stacks connected in series using a right greedy algorithm (see [221] for more details). West-2-stack-sortable permutations do not form a class, nevertheless, it is conceivable to characterize them using a few kinds of generalized patterns (barred patterns).

Imposing some restrictions on the content of the stack is considering another possible variation on the two-stacks problem. In [194], Rebecca Smith has studied the case in which the first stack is required to be decreasing.

Take note that, the second stack turns out to be necessarily increasing if we don't select a particular algorithm in advance. Within the case above, Smith can describe an optimal sorting algorithm, thanks to which she can completely characterize (in terms of avoided patterns) and enumerate sortable permutations.

4.2 Sorting with Restricted Stacks

Since the (classical) problem of characterizing and enumerating permutations that can be sorted using two stacks connected in series is still largely open, Cerbai and al. presented in [60] a related problem, in which they imposed restrictions both on the procedure and on the stacks. More precisely, they considered a greedy algorithm where they perform the rightmost legal operation (here "rightmost" refers to the usual representation of stack sorting problems).

In this thesis, we will deal with similar sorting machines consisting of two stacks connected in series (see Fig. 4.4). Recalling the key properties of the Stack-sort algorithm, we will consider machines obeying certain constraints, which are described below.

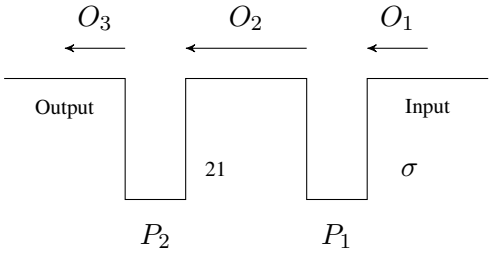


Figure 4.4: The σ-machine

1. The stacks must obey some restrictions, which are expressed by saying that, at each step of the execution, the elements into each stack (read from top to bottom) must avoid certain forbidden configurations. In particular, in analogy with Stack-sort, we require the second stack to be increasing. Notice that this can be equivalently expressed as follows: at every step, the sequence of numbers contained in the stack (read from top to bottom) has to avoid pattern 21. We will express this by saying that the stack is 21-avoiding. Moreover, we will be interested in machines in which the first stack is σ-avoiding, for some pattern σ.

2. The algorithm we perform on the two stacks connected in series is right greedy. As already observed, this is equivalent to making two passes through a stack, performing the right greedy algorithm at each pass. However, due to the restriction described above, during the first pass, the stack is σ-avoiding, whereas during the second pass it is 21-avoiding.

We will use the term σ-machine to refer to the right greedy algorithm performed on two stacks in series, such that the first stack is σ-avoiding and the second stack is 21-avoiding. Formally, the algorithm we are going to analyze is described in the Listing below. The set of permutations which are sortable by the σ-machine is denoted $Sort(\sigma)$ and its elements are the σ-sortable permutations.

The set of σ-sortable permutations of length n is denoted $Sort_n(\sigma)$.

$Stack_\sigma := \emptyset$;
$Stack_I := \emptyset$;
$i := 1$;
while $i \leq n$ **do**
  if $\sigma \not\leq Stack_\sigma \circ w_i$ **then**
    execute S_σ;
    $i := i + 1$;
  end
  else if $Stack_I = \emptyset$ *or* $TOP(Stack_\sigma) < TOP(Stack_I)$ **then**
    execute S_I;
  end
  execute O;
end
while $Stack_\sigma \neq \emptyset$ **do**
  if $Stack_I = \emptyset$ *or* $TOP(Stack_\sigma) < TOP(Stack_I)$ **then**
    execute S_I;
  else
    execute O;
  end
end
while $Stack_I \neq \emptyset$ **do**
  execute O;
end

Algorithm 4.1: The σ-machine ($Stack_\sigma$ is the σ-avoiding stack, $Stack_I$ is the increasing stack, S_σ means pushing into $Stack_\sigma$, S_I means pushing into $Stack_I$, O means moving TOP($Stack_I$) into the output, $\circ$ is the concatenation operation)

Let P be a set of patterns. A P-stack is a stack that is not allowed to contain an occurrence of any pattern in P, reading its elements from top to bottom. Given a permutation π, denote by $\mathrm{out}^T(\pi)$ the permutation obtained after passing the permutation w through the P-avoiding stack by applying a greedy procedure, i.e. by always pushing the next element of the input, unless it creates an occurrence of a forbidden pattern inside the stack. Denote by $\mathrm{Sort}_n(P)$ the set of length n permutations that are sortable by the P-machine, that is, by passing w through the P-avoiding stack and then through the 21-avoiding stack. Permutations in $\mathrm{Sort}_n(P)$ are called *P-sortable*, and $\mathrm{Sort}(P)$ is the set of P-sortable permutations of any length. As a consequence of Proposition 1, $\mathrm{Sort}(P)$ consists precisely of those permutations w for which $\mathrm{out}^P(w)$ avoids 231. To ease notations, if P is either a singleton $P = \{\sigma\}$ or a pair of patterns $P = \{\sigma, \tau\}$, we will omit the curly brackets from the above notations. For instance, we will write $\mathrm{Sort}(\sigma, \tau)$ instead of $\mathrm{Sort}(\{\sigma, \tau\})$.

There were many works on the pattern-avoiding machine. For instance, besides introducing the machine, the authors of [60] first proved that the set of σ-machines whose associated sortable permutations are not a class is counted by Catalan numbers. Moreover, they analyzed two specific σ-machines in full detail (namely when $\sigma = 321$ and $\sigma = 123$), providing for each of them a complete characterization and enumeration of sortable permutations. Also, they showed that if w is a 12-sortable permutation of length n, then $\mathrm{out}^{12}(w) = n(n-1)\dots1$. Moreover, by Proposition 1 and applying the complement operation on the processed permutation, we have that

$\mathrm{Sort}(12) = \mathfrak{S}(213)$. To refer to this result later, we state it below in a slightly more general form. A *partial permutation* of n is an injection $w : \{1, 2, \ldots, k\} \to \{1, 2, \ldots, n\}$, for some $0 \leq k \leq n$, and the integer k is said to be the length of w. We let a 12-stack act on a partial permutation w of n in a natural way by identifying w with the list of its images.

Proposition 2. If w is a partial permutation of n which is 12-sortable, then $\mathrm{out}^{12}(w)$ is the decreasing rearrangement of the symbols of w. Moreover, w is 12-sortable if and only if it avoids 213.

Then, in [59], Cerbai generalized these Pattern-avoiding Machines by allowing permutations with repeated elements, also known as Cayley permutations. The main result is a description of those patterns such that the corresponding set of sortable permutations is a class. They also showed a new involution on the set of Cayley permutations, obtained by regarding a pattern-avoiding stack as an operator. Also, they analyzed two generalizations of pop-stack sorting on Cayley permutations. In both cases, we describe sortable permutations in terms of pattern avoidance.

Finally, in [61], Cerbai et al. continued the analysis of the pattern-avoiding sorting machines introduced by Cerbai, Claesson and, Ferrari [60]. They characterized and enumerated the set of permutations that can be sorted when the first stack is 132-avoiding, solving one of the open problems proposed in [60]. To that end, they presented several connections with other well-known combinatorial objects, such as lattice paths and restricted growth functions (which encode set partitions). They also provided new proofs for the enumeration of some sets of pattern-avoiding restricted growth functions and they expect that the tools introduced can be fruitfully employed to get further similar results.

4.3 Permutations sortable by the (σ, τ)-machine

In this thesis, we study a variant of pattern-avoiding machines where the first stack avoids (σ, τ), a pair of patterns of length three. Following [60], we call it (σ, τ)-machine. More specifically, we restrict ourselves to those pairs of patterns for which sortable permutations are counted by either the Catalan numbers or two of their close relatives: the binomial transform of Catalan numbers and the Schröder numbers. For the pair $(132, 231)$ we show that sortable permutations are those avoiding 1324 and 2314, a set whose enumeration is given by the large Schröder numbers. Under certain conditions on the avoided patterns, the output of the first stack is bijectively related to its input (see [33, 59]): it follows that for three pairs of patterns, namely $(123, 213)$, $(132, 312)$ and $(231, 321)$, sortable permutations are counted by the Catalan numbers. This result was proved independently in [21, 33]. For the pair $(123, 132)$, we prove that sortable permutations are those avoiding the patterns 2314, 3214, 4213 and the generalized pattern $[24\bar{1}3$ (a permutation w avoids the pattern $[24\bar{1}3$ if for any subsequence $w_1 w_i w_j$, with $1 < i < j$ and $w_1 < w_j < w_i$, there is an index t, $i < t < j$, such that $\pi_1 \pi_i \pi_t \pi_j$ is an occurrence of 2413). We prove that sortable permutations are enumerated by the Catalan numbers by showing that the distribution of the first element is given by the well-known Catalan triangle. Finally, we show that for the pair $(123, 312)$ the corresponding counting sequence is the binomial transform of Catalan numbers.

An entry w_i of a permutation w is a *left-to-right minimum* if $w_i < w_j$, for each $j < i$. The *left-to-right minima decomposition* (briefly *ltr-min decomposition*) of w is $w = m_1 B_1 m_2 B_2 \cdots m_t B_t$, where $m_1 > m_2 > \cdots > m_t$ are the ltr-minima of w and the *block* B_i contains the elements of w between m_i and m_{i+1}, for $i = 1, \ldots, t-1$. The last block B_t contains the elements that follow m_t in w. Note that $m_t = 1$. The notion of a *left-to-right maximum* of a permutation

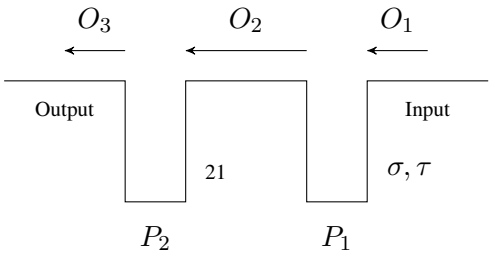


Figure 4.5: The (σ, τ)-machine consists of two stacks in series where the first stack P_1 avoids (from top to bottom) σ and τ while the second P_2 avoids the pattern 21. At each step of the process, we perform the rightmost possible operation among O_1, O_2, O_3, where O_1 pushes in P_1 the current entry of the input permutation, O_2 pops the top of P_1 and pushes it in P_2, and O_3 pops the top of P_2 and pushes it in the output permutation. For instance, if $\sigma = 123$, $\tau = 132$ then $\pi = 35124$ is sortable by applying the following operations: $O_1, O_1, O_2, O_1, O_1, O_1, O_2, O_2, O_2, O_3, O_3, O_2, O_3, O_3, O_3$.

w is defined similarly. The *ltr-max decomposition* of w is $w = M_1 B_1 M_2 B_2 \cdots M_t B_t$, where $M_1 < M_2 < \cdots < M_t$ are the ltr-maxima of w. In this case, $M_t = n$, where n is the length of w.

4.3.1 Pair $(132, 231)$

This section is devoted to the analysis of the $(132, 231)$-machine.

Theorem 1. Consider the $(132, \sigma)$-machine, where $\sigma = \sigma_1 \cdots \sigma_{k-1}\sigma_k \in \mathfrak{S}_k$, with $k \geq 3$ and $\sigma_{k-1} > \sigma_k$. Given a permutation π of length n, let $m_1 B_1 \cdots m_t B_t = \pi$ be its ltr-min decomposition. Then:

1. Every time an ltr-minimum m_i is pushed into the $(132, \sigma)$-stack, the $(132, \sigma)$-stack contains the elements $m_{i-1}, \ldots, m_2, m_1$, reading from top to bottom. Moreover, we have
$$\mathrm{out}^{132,\sigma}(\pi) = \tilde{B}_1 \cdots \tilde{B}_t m_t \cdots m_1,$$
where $\tilde{B}_i$ is a rearrangement of B_i.
2. If π is $(132, \sigma)$-sortable, then $\tilde{B}_i$ is decreasing for each i. Moreover, for each $i \leq t-1$, we have $B_i > B_{i+1}$ (i.e. $x > y$ for each $x \in B_i, y \in B_{i+1}$).

Proof. 1. Let us consider the evolution of the $(132, \sigma)$-stack on input π. Note that, since $k \geq 3$, the element m_1 remains at the bottom of the $(132, \sigma)$-stack until the end of the process. Now, if B_1 is not empty then for each $x \in B_1$, the elements $m_2 x m_1$ form an occurrence of 132. Therefore the block B_1 is extracted before m_2 enters the $(132, \sigma)$-stack. After m_2 is pushed, the $(132, \sigma)$-stack contains $m_2 m_1$, reading from top to bottom. Since $m_2 < m_1$, but $\sigma_{k-1} > \sigma_k$ by hypothesis, m_2 cannot play the role of either σ_{k-1} in an occurrence of σ or of 3 in an occurrence of 132. Thus m_2 remains at the bottom of the $(132, \sigma)$-stack until the end of the sorting procedure. The thesis follows by iterating the same argument on each block B_i, for $i \geq 2$.

2. Suppose that π is $(132, \sigma)$-sortable. Assume, for a contradiction, that $\tilde{B}_i$ is not decreasing, for some i. Then there are two consecutive elements $x < y$ in $\tilde{B}_i$. Therefore, by what was proved above, $\mathrm{out}^{132,\sigma}(\pi)$ contains an occurrence xym_t of 231, which is impossible due to Proposition 1. Finally, suppose that $x < y$, for $x \in B_i$ and $y \in B_{i+1}$. Then xym_t is an occurrence of 231 in $\mathrm{out}^{132,\sigma}(\pi)$, a contradiction.

□

Theorem 1 and Proposition 2 guarantee that if $\pi = m_1 B_1 \cdots m_t B_t$ is the ltr-min decomposition of a $(132, 231)$-sortable permutation π, then (with the notation above) $\tilde{B}_i = \mathrm{out}^{12}(B_i)$, for each i. However, this is true even when the sortability requirement is relaxed.

Lemma 4. *Let $\pi = m_1 B_1 \cdots m_t B_t$ be the ltr-min decomposition of a permutation π. Write $\mathrm{out}^{132,231}(\pi) = \tilde{B}_1 \cdots \tilde{B}_t m_t \cdots m_1$ as in Theorem 1. Then $\tilde{B}_i = \mathrm{out}^{12}(B_i)$, for each i.*

Proof. Consider the instant immediately after m_i is pushed into the $(132, 231)$-stack and the non-empty block B_i has to be processed, for some i. By Theorem 1, at this point the $(132, 231)$-stack contains $m_i, m_{i-1}, \ldots, m_1$, reading from top to bottom. We want to show that the behavior of the $(132, 231)$-stack on B_i is equivalent to the behavior of an empty 12-stack on input B_i. We prove that the $(132, 231)$-stack performs the pop operation of some $x \in B_i$ if and only if the 12-stack does the same. If either the next element of the input is m_{i+1} or x is the last element of π to be processed, then both the $(132, 231)$-stack and the 12-stack perform a pop operation, as desired. Otherwise, suppose the next element of the input is y, for some y in the same block B_i, and the $(132, 231)$-stack pops the element $x \in B_i$. This means that the $(132, 231)$-stack contains two elements z, w, with z above w, such that yzw is an occurrence of either 132 or 231. Note that, since $z > w$, z is not an ltr-minimum. Therefore yz is an occurrence of 12 and the 12-stack performs a pop operation, as desired. Conversely, suppose that the 12-stack pops the element x, with $y \in B_i$ the next element of the input. This implies that the 12-stack contains an element z such that $z > y$. Therefore yzm_i is an occurrence of 231 and the $(132, 231)$-stack performs a pop operation, as desired.

□

Corollary 1. Let $\pi = m_1 B_1 \cdots m_t B_t$ be the ltr-min decomposition of a permutation π. Then the following are equivalent.

1. B_i avoids 213 and $B_i > B_{i+1}$, for each i.
2. π is $(132, 231)$-sortable.
3. $\pi \in \mathtt{Av}(1324, 2314)$.

Proof. Combining the first point in Theorem 1 and Lemma 4 we have:

$$\mathrm{out}^{132,231}(\pi) = \mathrm{out}^{12}(B_1) \cdots \mathrm{out}^{12}(B_t) m_t \cdots m_1.$$

We will use this decomposition of $\mathrm{out}^{132,231}(\pi)$ throughout the rest of the proof.

$[1 \Rightarrow 2]$ Suppose, for a contradiction, that $\mathrm{out}^{132,231}(\pi)$ contains an occurrence bca of 231. Note that, since $c > a$, while $m_t < \cdots < m_1$, c is not a ltr-minimum of π (and thus neither b is). Now, if b and c are in the same block B_j, then $\mathrm{out}^{12}(B_j)$ is not decreasing. Thus, by Proposition 2, B_j contains 213, which is a contradiction. Otherwise, if $b \in B_j$ and $c \in B_k$, with $j < k$, then we have a contradiction with the hypothesis $B_i > B_{i+1}$ for each i.

$[2 \Rightarrow 3]$ Suppose, for a contradiction, that $\pi \notin \mathtt{Av}(1324, 2314)$. First, suppose that π contains an occurrence $acbd$ of 1324. Observe that b, c, d are not ltr-minima of π. Now, if b and d are in the same block B_j of π, for some j, then B_j contains an occurrence cbd of 213. Therefore $\mathrm{out}^{12}(B_j)$

contains an occurrence of 231 due to Proposition 2, which contradicts the hypothesis. Otherwise, if $b \in B_j$ and $d \in B_k$, for some $j < k$, then $\mathrm{out}^{132,231}(\pi)$ contains an occurrence bdm_k of 231, again a contradiction. The pattern 2314 can be addressed analogously, so we leave it to the reader.

$[3 \Rightarrow 1]$ Let $\pi \in \mathtt{Av}(1324, 2314)$. If B_i contains an occurrence bac of 213, then π contains an occurrence $m_i bac$ of 1324, which is impossible. Otherwise, if π contains two elements $x \in B_j$, $y \in B_k$, with $x < y$ and $j < k$, then $m_j x m_k y$ is an occurrence of 2314, contradicting the hypothesis. □

The enumeration of $\mathtt{Av}(1324, 2314)$ (or symmetry of these patterns) can be found for instance in [17, 223]. Note that in [15], the authors provide a constructive bijection between these permutations and Schröder paths.

Corollary 2. Permutations of length n in $Sort(132, 231)$ are enumerated by the large Schröder numbers (sequence `A006318` in [2]).

4.3.2 The $(\sigma, \hat{\sigma})$-machine

For a permutation σ of length two or more, denote by $\hat{\sigma}$ the permutation obtained from σ by interchanging its first two entries. Let us regard a (σ,τ)-stack as an operator $\mathrm{out}^{\sigma,\tau} : \mathfrak{S} \to \mathfrak{S}$. By conveniently modifying the proof of Corollary 4.5 in [59] (stated in the context of Cayley permutations), we have that $\mathrm{out}^{\sigma,\tau}$ is a length preserving bijection on $\mathfrak{S}$ if and only if $\tau = \hat{\sigma}$. More generally, Berlow [33] showed that for a set P of patterns, out^{T} is a length preserving bijection on $\mathfrak{S}$ if and only if P is closed under the $\hat{}$ operator. For the paper to be self-contained, we shall give the following result, which is easier to prove (although weaker): $\mathrm{out}^{\sigma,\hat{\sigma}}$ is a bijection for any pattern σ. An immediate consequence will be Theorem 2 below.

Let N^* be the set of finite length integer sequences. The action of the (σ,τ)-stack on input π can be naturally represented as a sequence of triples $(r; s; t) \in (N^*)^3$, where r is the current content of the output, s is the current content of the (σ,τ)-stack (read from top to bottom) and t is the current content of the input. The triple $(r; s; t)$ is said to be a *state of the passing* of π through the (σ,τ)-stack. Clearly, r is a prefix of $\mathrm{out}^{\sigma,\tau}(\pi)$, t is a suffix of π, the initial state is $(\lambda; \lambda; \pi)$ and the final one is $(\mathrm{out}^{\sigma,\tau}(\pi); \lambda; \lambda)$, where λ is the empty sequence. Moreover, a non-final state $(p_1 p_2 \cdots p_a; s_1 s_2 \cdots s_b; t_1 t_2 \cdots t_c)$ is followed by either the state

$$(p_1 p_2 \cdots p_a s_1; s_2 \cdots s_b; t_1 t_2 \cdots t_c),$$

if a pop operation is performed next, or

$$(p_1 p_2 \cdots p_a; t_1 s_1 s_2 \cdots s_b; t_2 \cdots t_c),$$

if a push operation is performed next.

For $p = p_1 \cdots p_n \in \mathbb{N}^n$, we denote by p^{r} the *reverse* of p, that is $p^{\mathrm{r}} = p_n \cdots p_1$. We wish to show that the behavior of the $(\sigma,\hat{\sigma})$-stack on π is strictly related to its behavior on $\left(\mathrm{out}^{\sigma,\hat{\sigma}}(\pi)\right)^{\mathrm{r}}$. More precisely, if $o_1 \cdots o_{2n}$ is the sequence of push/pop operations performed when π is passed through a $(\sigma,\hat{\sigma})$-stack, then $o'_{2n} \cdots o'_1$ is the sequence of push/pop operations performed when $\left(\mathrm{out}^{\sigma,\hat{\sigma}}(\pi)\right)^{\mathrm{r}}$ is passed through the $(\sigma,\hat{\sigma})$-stack, where o'_i is a push (resp. pop) operation if o_i is a pop (resp. push) operation. This can be equivalently expressed by saying that the state $(p; s; t)$ is followed by $(u; v; w)$ if and only if the state $(w^{\mathrm{r}}; v; u^{\mathrm{r}})$ is followed by $(t^{\mathrm{r}}; s; p^{\mathrm{r}})$.

Lemma 5. *Consider the action of the $(\sigma, \hat{\sigma})$-stack. Let $p, s, t \in \mathbb{N}^*$ and $x \in \mathbb{N}$.*

1. *If the state (p, xs, t) is followed by the state (px, s, t) (and thus a pop operation is performed) then the state $(t^{\mathrm{r}}, s, xp^{\mathrm{r}})$ is followed by the state $(t^{\mathrm{r}}, xs, p^{\mathrm{r}})$ (and thus a push operation is performed).*
2. *If the state (p, s, xt) is followed by the state (p, xs, t) (and thus a push operation is performed), then the state $(t^{\mathrm{r}}, xs, p^{\mathrm{r}})$ is followed by the state $(t^{\mathrm{r}}x, s, p^{\mathrm{r}})$ (and thus a pop operation is performed).*

Proof. 1. Since xs is the content of the $(\sigma, \hat{\sigma})$-stack in the state (p, xs, t), we have that xs avoids σ and $\hat{\sigma}$. Thus a push operation is performed if s is the content of the $(\sigma, \hat{\sigma})$-stack and x is the next element of the input.

2. If p is empty, the statement holds. Otherwise, let $p = p_1 \cdots p_a$ and $s = s_1 \cdots s_b$. Observe that p_a is the last element that has been extracted from the $(\sigma, \hat{\sigma})$-stack before x enters. Therefore, when p_a is extracted, p_a plays the role of either σ_2 in an occurrence of σ or of $\hat{\sigma}_2$ in an occurrence of $\hat{\sigma}$. More precisely, one of the following four cases holds. We show the details for the first case only, the others being similar. Let z be the length of σ.
   - $s_\ell p_a s_{i_3} \cdots s_{i_z}$ is an occurrence of σ, for some $\ell \geq 1$ and $\ell < i_3 < \cdots < i_z$. Then $p_a s_\ell s_{i_3} \cdots s_{i_z}$ is an occurrence of $\hat{\sigma}$ and therefore a pop operation is performed when p_a is the next element of the input and xs is the content of the $(\sigma, \hat{\sigma})$-stack, as desired.
   - $s_\ell p_a s_{i_3} \cdots s_{i_z}$ is an occurrence of $\hat{\sigma}$, for some $\ell \geq 1$ and $\ell < i_3 < \cdots < i_z$.
   - $x p_a s_{i_3} \cdots s_{i_z}$ is an occurrence of σ, for some $i_3 < \cdots < i_z$.
   - $x p_a s_{i_3} \cdots s_{i_z}$ is an occurrence of $\hat{\sigma}$, for some $i_3 < \cdots < i_z$.

□

A straightforward consequence of the previous lemma is that the map $\left(\mathrm{out}^{\sigma,\hat{\sigma}}\right)^{\mathrm{r}} : \mathfrak{S} \to \mathfrak{S}$ is its own inverse, and thus a bijection. More specifically, for any permutation π, we have $\left(\mathrm{out}^{\sigma,\hat{\sigma}}\right)^{-1}(\pi) = \left(\mathrm{out}^{\sigma,\hat{\sigma}}(\pi^{\mathrm{r}})\right)^{\mathrm{r}}$. Since π is $(\sigma, \hat{\sigma})$-sortable if and only if $\mathrm{out}^{\sigma,\hat{\sigma}}(\pi)$ avoids 231 (and the reverse map is bijective), we have that $\mathrm{Sort}(\sigma, \hat{\sigma})$ is in bijection with $\mathtt{Av}(231)$. The next theorem follows.

Theorem 2. For any pattern σ, $\mathrm{out}_n^{\sigma,\hat{\sigma}}$ is a bijection on $\mathfrak{S}_n$. Moreover, we have $|\mathrm{Sort}_n(123, 213)| = |\mathrm{Sort}_n(132, 312)| = |\mathrm{Sort}_n(231, 321)| = c_n$, the *nth* Catalan number.

4.3.3 Pair $(123, 132)$

We characterize $\mathrm{Sort}(123, 132)$ in terms of pattern avoidance. Then we show that $(123, 132)$-sortable permutations are enumerated by the Catalan numbers by exhibiting a link with the very well studied Catalan triangle.

Theorem 3. A permutation π is $(123, 132)$-sortable if and only if π avoids 2314,3214, 4213 and $[24\bar{1}3$.

Proof. □

Proposition 3. The distribution of the first element in $\mathrm{Sort}(123, 132)$ is given by the Catalan triangle (sequence A009766 in [2]).

Proof. Let $A_n(k)$ be the set of $(123, 132)$-sortable permutations of length n and starting with k. Let $A_n^1(k)$ be the subset of $A_n(k)$ consisting of those permutations $\pi = \pi_1\pi_2\dots\pi_n$ where any occurrence $\pi_1\pi_i\pi_\ell$ of $[231$ with $\pi_\ell = \pi_1 - 1$ can be extended into an occurrence $\pi_1\pi_i\pi_j\pi_\ell$ of $[3412$. Set $A_n^2(k) = A_n(k)\backslash A_n^1(k)$ and let $k \geq 2$. We shall provide bijections $\alpha : A_n^1(k) \to A_n(k-1)$ and $\beta : A_n^2(k) \to A_{n-1}(k)$.

Define $\alpha : A_n^1(k) \to A_n(k-1)$ by $\alpha(\pi) = \pi'$, where π' is obtained from π by swapping the two entries π_1 and $\pi_1 - 1$ in π. Since $\pi \in A_n^1(k)$, it is easy to check that π' avoids $[2\bar{4}13$. In addition, swapping π_1 and $\pi_1 - 1$ does not affect the avoidance of the three patterns 3214, 2314, 4213, which implies (see Theorem 3) that $\alpha(\pi) \in A_n(k-1)$.

Next define $\beta : A_n^2(k) \to A_{n-1}(k)$ by $\beta(\pi) = \pi''$, where π'' is obtained from π by deleting the entry π_ℓ immediately before $k-1$ and by decreasing by one all entries of π greater than π_ℓ. Notice that $\beta(\pi) \in A_{n-1}(k)$. Let us now sketch the proof that β is bijective, leaving some technical details to the reader. We shall explicitly define the inverse map of β. Given $\pi \in A_{n-1}(k)$, choose an integer ℓ as follows:

- ℓ is the minimal entry $l = \pi_u > \pi_1$, with $1 < u < i$, such that there is an index v with $\pi_v < \pi_i$ and $u < v < i$, if such entry exists.
- Otherwise, set $\ell = n$.

The preimage π is obtained by inserting ℓ immediately before $\pi_i = k - 1$ and then increasing by one all the entries π_j of π with $\pi_j \geq \ell$.

Finally, setting $a_n^k = |A_n(k)|$, we have that $a_n^k = a_n^{k-1} + a_{n-1}^k$, for $2 \leq k \leq n$. Since $A_n(1) = \{123\cdots n\}$ and $A_n(n)$ is the set of length n permutations avoiding 213 and starting with n, the initial conditions are given by $a_n^1 = 1$ and $a_n^n = c_{n-1}$, where c_n is the nth Catalan number. Therefore, a_n^k generates the well-known Catalan triangle (see Table 4.1 and [56, 78, 192]). □

$k\backslash n$	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1
2		1	2	3	4	5	6	7
3			2	5	9	14	20	27
4				5	14	28	48	75
5					14	42	90	165
6						42	132	297
...							...	...
$\sum$	1	2	5	14	42	132	429	1430

Table 4.1: The Catalan triangle $a_n^k = |A_n(k)|$, with $1 \leq n \leq 8$ and $1 \leq k \leq 6$.

Corollary 3. Permutations of length n in $\mathrm{Sort}(123, 132)$ are enumerated by the Catalan numbers.

Proof. With the previous notations we have $|\mathrm{Sort}_n(123, 132)| = \sum_{k=1}^n a_n^k = c_n$, the nth Catalan number (see again [56, 78, 192]). □

4.3.4 Pair $(123, 312)$

We start by giving a ltr-max counterpart of Theorem 1.

Theorem 4. Consider the $(312, \sigma)$-machine, where $\sigma = \sigma_1 \cdots \sigma_{k-1}\sigma_k \in \mathfrak{S}_k$, with $k \geq 3$ and $\sigma_{k-1} < \sigma_k$. Given a permutation π of length n, let $\pi = M_1 B_1 \cdots M_t B_t$ be its ltr-max decomposition. Then:

1. Everytime a ltr-maximum M_i is pushed into the $(312, \sigma)$-stack, the $(312, \sigma)$-stack contains the elements $M_i, M_{i-1}, \ldots, M_2, M_1$, reading from top to bottom. Moreover, we have
$$\mathrm{out}^{312,\sigma}(\pi) = \tilde{B}_1 \cdots \tilde{B}_t M_t \cdots M_1,$$
where $\tilde{B}_i$ is a rearrangement of B_i.
2. If π is $(312, \sigma)$-sortable, then $M_1, M_2, \ldots, M_t = n - t + 1, n - t + 2, \ldots, n$.

Proof. 1. The proof is identical to that of the first part of Theorem 1.

2. If π is $(312, \sigma)$-sortable, then $\mathrm{out}^{312,\sigma}(\pi)$ avoids 231 by Proposition 1. Suppose, for a contradiction, that there is an element $j \in \{n - t + 1, \ldots, n\}$ which is not a ltr-maximum. Note that $j \neq \pi_1 = M_1$ and $j \neq n = M_t$. Then, $\mathrm{out}^{312,\sigma}(\pi)$ contains an occurrence jnM_1 of 231, which is a contradiction. □

Instantiating σ by 123 in the previous theorem we have the next result.

Theorem 5. Let π be a $(123, 312)$-sortable permutation and let $\pi = M_1 B_1 \cdots M_t B_t$ be its ltr-max decomposition. Then:

1. B_i avoids 213 for each i.
2. $\tilde{B}_i = \mathrm{out}^{12}(B_i)$, for each i.

Proof. Let $i \geq 2$. Notice that, as a consequence of Theorem 4, immediately after M_i has been pushed in the $(123, 312)$-stack, this stack contains the elements $M_i \cdots M_2 M_1$, reading from top to bottom. Moreover, these elements remain at the bottom of the $(123, 312)$-stack until the end of the sorting procedure, since they are the last elements of $\mathrm{out}^{123,312}(\pi)$. This fact will be used for the rest of the proof.

1. Suppose, for a contradiction, that B_i contains an occurrence of 213, for some i, and let bac be such an occurrence with a 'minimal', in the sense that there is no $a' < a$ where $ba'c$ is an occurrence of 213 in B_i. Therefore, since abM_i is an occurrence of 123, b is extracted from the $(123, 312)$-stack before a enters. In addition, when c enters into the $(123, 312)$-stack, a is still in this stack. Indeed, no entry in B_i between a and c together with a produces a forbidden pattern in the $(123, 312)$-stack. It follows that $\mathrm{out}^{123,312}(\pi)$ contains bca which is an occurrence of 231, yielding a contradiction with the sortability of π.

2. Let us consider the action of the $(123, 312)$-stack on block B_i. We wish to show that the behavior of the $(123, 312)$-stack when processing B_i is equivalent to the behavior of an empty 12-stack on input B_i. In other words, we prove that the restriction of the $(123, 312)$-stack is

triggered if and only if the next element of the input forms an occurrence of 12 together with some other element in the $(123, 312)$-stack. Immediately after M_i has been pushed (i.e. before the first element of B_i is processed), the $(123, 312)$-stack contains the elements $M_i \cdots M_2 M_1$, reading from top to bottom. Observe that B_i avoids 213 by what was proved above, therefore the $(123, 312)$-stack cannot be triggered by an occurrence of 312 when processing B_i. Suppose that the next element of the input x forms an occurrence xy of 12 with some $y \in B_i$. Then xyM_i is an occurrence of 123 in the $(123, 312)$-stack, and so this stack behaves as a 12-stack. Conversely, suppose that the $(123, 312)$-stack is triggered by an occurrence of xyz of 123, where x is the next element of the input. Since $M_i > M_{i-1} > \cdots > M_1$, necessarily $y \in B_i$. Thus xy is an occurrence of 12 that triggers the 12-stack, as wanted. □

As a consequence of what proved so far in this section, for any $(123, 312)$-sortable permutation $\pi = M_1 B_1 \cdots M_t B_t$ of length n, we have $B_i \in \mathtt{Av}(213)$ and $M_1, \ldots, M_t = n - t + 1, \ldots, n$. Moreover, by Proposition 2, each $\tilde{B}_i$ in $\mathrm{out}^{123,312}(\pi) = \tilde{B}_1 \cdots \tilde{B}_t M_t \cdots M_1$ is decreasing. Therefore, for any three elements x, y, z, with $x \in B_i$, $y \in B_j$ and $z \in B_k$, with $i < j \leq k$, xyz is not an occurrence of 231. Otherwise, xyz would still be an occurrence of 231 in $\mathrm{out}^{123,312}(\pi)$, contradicting the fact that π is $(123, 312)$-sortable. From now on, we say that xyz is an occurrence of $2 - 3 - 1$ if $z < x < y$, with $x \in B_i$, $y \in B_j$ and $z \in B_k$, with $i < j < k$. Similarly, when $j = k$, we say that xyz is an occurrence of $2 - 31$.

Theorem 6. Let $\pi = M_1 B_1 \cdots M_t B_t$ be the ltr-max decomposition of a permutation π of length n. Write $\mathrm{out}^{123,312}(\pi) = \tilde{B}_1 \cdots \tilde{B}_t M_t \cdots M_1$ as in Theorem 4. Then π is $(123, 312)$-sortable if and only if the following conditions are satisfied:

1. $M_j = n - t + j$, for each $j = 1, \ldots, t$.
2. B_i avoids 213 for each i (and thus $\tilde{B}_i$ is decreasing for each i).
3. $\mathrm{out}^{123,312}(\pi)$ avoids $2 - 3 - 1$.
4. $\mathrm{out}^{123,312}(\pi)$ avoids $2 - 31$.

Proof. If π is $(123, 312)$-sortable, then π satisfies all the above conditions as a consequence of what was proved before in this section. Conversely, it is easy to check that if π satisfies the above conditions, then $\mathrm{out}^{123,312}(\pi)$ avoids 231. Thus π is $(123, 312)$-sortable. □

Reformulating the third condition of Theorem 6 we obtain the following lemma, whose easy proof is omitted.

Lemma 6. *Let $\pi = M_1 B_1 \cdots M_t B_t$ be the ltr-max decomposition of the $(123, 312)$-sortable permutation π. Write $\mathrm{out}^{123,312}(\pi) = \tilde{B}_1 \cdots \tilde{B}_t M_t \cdots M_1$ as in Theorem 4. Then $\mathrm{out}^{123,312}(\pi)$ avoids $2 - 31$ if and only if for each $x \in B_i$, $y \in B_j$, with $i < j$, we have:*

- *if $y > x$, then $B_j > x$.*
- *If $y < x$, then $B_j < x$.*

In other words, Lemma 6 says that each block B_j of a $(123, 312)$-sortable permutation π is bounded between two previous elements of π. The following result is obtained by restating this lemma and Theorem 6 in terms of pattern avoidance.

Theorem 7. A permutation π is $(123, 312)$-sortable if and only if π avoids the three generalized patterns $[132$, $[42531$ and $[42\overline{1}53$.

Next we prove that $(123, 312)$-sortable permutations are enumerated by the binomial transform of Catalan numbers. We shall exploit the above characterization in terms of patterns in order to provide a bijection with a certain set of partial permutations, whose enumeration is straightforward.

Recall from Section 4.2 that a partial permutation of n is an injection $\pi : \{1, 2, \ldots, k\} \to \{1, 2, \ldots, n\}$, for some $0 \leq k \leq n$. The partial permutation of length zero will be denoted by λ. Denote by $\mathcal{A}_n$ the set of all partial permutations of n. For instance, we have $\mathcal{A}_3 = \{\lambda, 1, 2, 3, 12, 21, 13, 31, 23, 32, 123, 132, 213, 231, 312, 321\}$. There is a natural bijection between the set of permutations in $\mathfrak{S}_n$ avoiding the pattern $[132$ and $\mathcal{A}_{n-1}$. Indeed, from a length n permutation π avoiding $[132$, we associate the unique partial permutation $\alpha(\pi) \in \mathcal{A}_{n-1}$ defined as follows:

$$\alpha(\pi)_{\pi_i} = i - 1, \text{ for } \pi_i < \pi_1.$$

In other words, $\alpha(\pi)$ is obtained by recording the indices (minus one) of the elements $\pi_i < \pi_1$, from the smallest to the largest one. For instance, if $\pi = 52461783$, then $\alpha(\pi) = 4172$. Notice that $\alpha(\pi) = \lambda$ if and only if $\pi_1 = 1$. Let us now define two pattern containments on $\mathcal{A}_n$. Let $a = a_1 a_2 \cdots a_m$ be a partial permutation of n, with $m \leq n$, and let $i < j < k$. Then $a_i a_j a_k$ is an occurrence of the pattern $31|2$ if it is an occurrence of 312 such that at least one value of the interval $[a_j, a_k]$ does not appear in a. Moreover, we say that $a_i a_j a_k$ is an occurrence of the pattern $\overline{2}1\overline{3}$ if it is an occurrence of 213 such that $a_i = a_k - 1$.[1] By interpreting Theorem 7 in terms of partial permutations, we obtain easily:

Theorem 8. A permutation π is $(123, 312)$-sortable if and only if $\alpha(\pi)$ avoids $31|2$ and $\overline{2}1\overline{3}$.

Let $\mathcal{A}_n(31|2, \overline{2}1\overline{3})$ be the set of partial permutations of n avoiding the two patterns $31|2$ and $\overline{2}1\overline{3}$, and $\mathcal{A}_n(213)$ be the set of partial permutations of n avoiding the classical pattern 213.

Theorem 9. For any $n \geq 1$, there is a bijection ϕ between $\mathcal{A}_n(31|2, \overline{2}1\overline{3})$ and $\mathcal{A}_n(213)$.

Proof. Let us define recursively the map ϕ from $\mathcal{A}_n(31|2, \overline{2}1\overline{3})$ to $\mathcal{A}_n(213)$. If $\pi = \lambda$, then we set $\phi(\pi) = \lambda$. Otherwise, π has a unique decomposition of the form $\pi = A \min(\pi) B$ where A and B are disjoint partial permutations of n. We distinguish three cases:

(i) If at least one of A or B is empty, then we set $\phi(\pi) = \phi(A) \min(\pi) \phi(B)$;

(ii) If both A and B are not empty and $\min(A) > \max(B)$, then we set $\phi(\pi) = \phi(A) \min(\pi) \phi(B)$. It is worth noting that the hypothesis that π avoids $31|2$ implies that any value $x \in [\min(\pi), \max(B)]$ occurs in B.

(iii) Suppose that both A and B are not empty and $\min(A) < \max(B)$. Since π avoids $\overline{2}1\overline{3}$, there exists $x \in [\min(A), \max(B)]$ such that x does not occur in π. We choose the smallest x with this property so that any value of the interval $[\min(A), x]$ occurs in A. Moreover, since π avoids $31|2$, it must be $\max(A) = x - 1$. An illustration of this case is depicted in Fig. 4.6. Let r be the maximum value of B that is lower than $\min(A)$ and consider the string B' obtained from B by decreasing by $x - r - 1$ all entries greater than x. Similarly,

[1] This is analogous to the notion of bivincular pattern on classical permutations

let A' be obtained from A by increasing by $\max(B) - x + 1$ all its entries. Obviously, A' and B' belong to $\mathcal{A}_n(31|2, \overline{2}1\overline{3})$, whereas $\pi' = A' \min(\pi) B'$ contains $31|2$. Then we set $\phi(\pi) = \phi(A') \min(\pi) \phi(B')$ (see again Fig. 4.6 for an illustration of this mapping). It is worth noting that the value $r + 1$ does not occur in both A' and B', which imply that there exists $y \in [\min(\pi), r + 1]$ such that y does not occur in $\phi(B')$.

Next we prove that ϕ is an injective map. We proceed by induction on the length of partial permutations. Let π be a partial permutation. Due to the remarks at the end of (ii) and (iii), the image of π under ϕ satisfying (ii) is a partial permutation π' such that any value $x \in [\min(\pi), \max(\phi(B))]$ occurs in $\phi(B)$, which is not true for a permutation π satisfying (iii). Then, for two non-empty partial permutations π and σ in $\mathcal{A}_n(31|2, \overline{2}1\overline{3})$, $\phi(\pi) = \phi(\sigma)$ implies that π and σ have the same length and they belong to the same case (i), (ii) or (iii). The recurrence hypothesis induces $\pi = \sigma$ which completes the induction.

Finally, observe that any partial permutation π avoiding 213 is of the form $A \min(\pi) B$ where $\min(A) > \max(B)$ and both A and B avoid 213. According to the geometrical shape of π (as in the proof of injectivity), π fits exactly in one of the cases (i), (ii) and (iii) in the definition of ϕ. Therefore the surjectivity of ϕ can be showed by using its recursive definition and induction on A and B. We leave the details to the reader. □

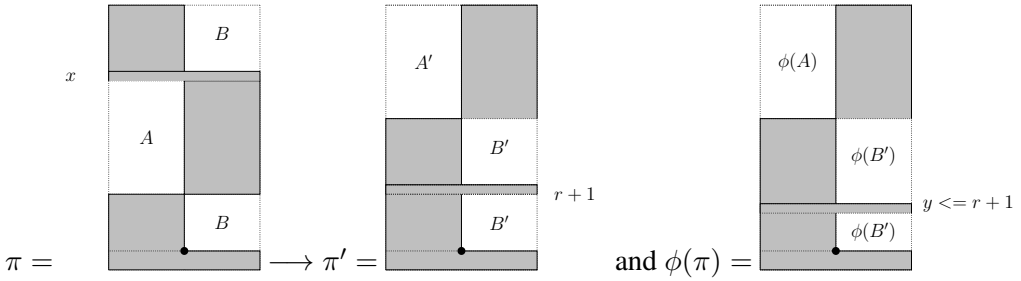


Figure 4.6: Illustration of ϕ in the case (iii) of the proof of Theorem 8.

Now, it is easy to enumerate the set $\mathcal{A}_n(213)$. Indeed any partial permutation $\pi \in \mathcal{A}_n(213)$ can be obtained by choosing k integers from $\{1, 2, \ldots, n\}$ and then arranging them according to the partial order of a permutation in $\mathtt{Av}_k(213)$ (there are c_k such permutations). Therefore, we have:

$$|\mathcal{A}_n(213)| = \sum_{k=0}^{n} \binom{n}{k} c_k,$$

the binomial transform of Catalan numbers (sequence `A007317` in [2]). The enumeration of $\mathrm{Sort}(123, 312)$ follows immediately.

Corollary 4. For each $n \geq 1$, we have:

$$|\mathrm{Sort}_{n+1}(123, 312)| = \sum_{k=0}^{n} \binom{n}{k} c_k.$$

4.4 Conclusion

In this chapter, we were interested in pattern avoiding machines where the first stack avoids a pair of patterns of length 3. They were introduced recently by Claesson, Cerbai and Ferrari as a particular case of the two-stacks in series sorting device. We investigated those pairs for which sortable permutations are counted by the (binomial transform of the) Catalan numbers and the Schröder numbers. We prove that for the pair $(132, 231)$ the enumeration of the set of sortable permutations is given by the large Schröder numbers; and for three pairs of patterns, $(123, 213), (132, 312)$ and $(231, 321)$, sortable permutations are counted by the Catalan numbers. Some of the obtained results have been further generalized to Cayley permutations by Cerbai, specialized to particular patterns by Defant and Zheng, or considered in the context of functions over the symmetric group by Berlow.

Part II

Evolutionary Optimization

Chapter 5

Evolutionary computation: an overview

The first part of this thesis focuses on enumerative combinatorics. It took into account the combinatorial properties of some combinatorial objects, in particular the properties of permutations. In the second part, as a complementary work, we study the application of permutation properties in evolutionary computation. In fact, the proposed contribution concerns permutation problems and genetic algorithms (more especially encoding and associated operators).

This chapter begins by presenting the permutation problems and why it is always interesting to design suitable methods for them, followed by the introduction and definition of evolutionary algorithms, genetic operators and indicators. Then, some optimization problems based on permutations are reviewed. The chapter ends with concluding remarks.

5.1 Permutation-based problems

Permutation-based optimization problems are a great example of the direct application of permutations and their properties since they use permutations to represent potential solutions. Applications of permutation problems can be found in various areas of real life, such as assignment applications which include a variety of design problems in different fields, production scheduling and sequencing applications, traveling salesman problems, routing issues, etc.

The mathematical properties of permutations could be useful for algorithms applied to this type of problem. Thus, in combinatorics, as defined in section 2.1.2, a permutation is an arrangement of a finite set of n objects denoted $[n] = \{1, 2, \ldots, n\}$, in a specific order and each element appears only once. There are exactly $n!$ possible permutations on n. A permutation π of length n can be represented as follows: $\pi_1\pi_2 \ldots \pi_n$. $\mathfrak{S}_n$, the set of all possible permutations of size n, forms a group with the composition of mappings as product and the identity permutation as a neutral element. The identity permutation is the permutation $\pi_1\pi_2 \ldots \pi_n$ such that: $\pi_1 < \pi_2 < \cdots < \pi_n$. For example, the identity permutation on the set $\{1, 2, 3, 4, 5\}$ is 12345.

Some examples of permutation-based problems in combinatorial optimization are summarized and classified by general application categories.

5.1.1 Assignment problems

Assignment problems are included in the category of optimization problems which consists in minimizing or maximizing a certain cost. They regroup problems of assignments between sets having the same size. Each problem is characterized by its size, the constraints on the assignments and the cost involved depending on the application domain. Burkard et al. provide in [51] a comprehensive review of assignment problems from their conceptual beginnings in the 1920s through theoretical, algorithmic, and practical developments.

The simplest problem within this category is the **Linear assignment problem** (LAP). Given two sets of objects of the same size n, that is to say a set of tasks to be executed on a set of machines. An assignment problem is said to be linear if the total cost of the assignment for all tasks is equal to the sum of the costs for each machine. The problem is to find an assignment p that optimizes the sum of all the assignments in pairs. Burkard and Cela in [50] have proposed a state of the art on linear assignment problems (LAP).

As an extension of linear assignment problems, Pierskalla [166] presented the **Multi-index assignment problems** (AP3) in 1968. First, only 3-index assignment problems have been considered, while in recent years, problems with more than 3 indices have been investigated [180–182].

The three index assignment problem (AP3), also introduced by Pierskalla [165], consists in assigning a set of p items to q locations at r points or intervals of time in such a way to minimize the cost of the assignment. It is assumed that $p \leq q \leq r$.

Only implicit enumeration methods are known for this type of problem, like the branch and bound methods [166, 218] or primal-dual implicit enumeration method based on a graph theoretic approach [112].

Also we have the **Quadratic assignment problem** (QAP), which was introduced by Koopmans and Beckmann in 1957 [129] as a mathematical model for the allocation of indivisible economical activities. It is one of the fundamental hardest combinatorial optimization problems in mathematics.

Formally, given two sets P and L of equal size n representing the sets of facilities and locations respectively. For each pair of locations, a distance $d : L \longrightarrow \mathbb{R}$ is specified and for each pair of facilities a weight w is indicated. The issue is to assign all facilities to different locations, $f : P \longrightarrow L$, with the objective of minimizing the cost function, which is the sum of the distances multiplied by the corresponding flows, see Eq. 5.1.

$$\sum_{a,b \in P} w(a,b) \cdot d(f(a), f(b)) \tag{5.1}$$

The QAP has been proved NP-hard by Sahni and Gonzalez [189], i.e., it cannot be solved in polynomial time, and gained growing attention within the optimization community because of its challenging complexity and the interesting number of applications in numerous real life domains such as scheduling, production, computer manufacture, chemistry, facility location, communication, and other fields.

5.1.2 Scheduling problems

Scheduling problems are widely encountered in real world applications mainly in industrial fields. The inputs of such problems are a set of jobs or tasks and a set of machines. The required output is a schedule – an assignment of jobs to machines. The schedule should optimize a certain objective function. Scheduling problems will be presented using permutations such as the flowshop and jobshop problems.

The **permutation flowshop problem**(FSP), introduced by Johnson in his paper [118], is one of the numerous scheduling problems. In the general flowshop problem (FSP), according to Brucker in [48], a set of unrelated n jobs $J_1, J_2, \ldots, J_n$ are to be processed on a set of m machines, $M_1, M_2, \ldots, M_m$. These machines are arranged in series and each job has to visit all of them in the same order. Therefore, each job J_i is composed of m consecutive tasks O_{ij}. The processing times of the jobs at the machines are known in advance, non-negative and deterministic. They are denoted by p_{ij} $(j = 1, \ldots, m)$ where O_{ij} means the j-th operation of the job J_i must be processed on machine M_j. The problem consists in obtaining a production sequence of jobs J_j , for each machine j, $(j = 1, \ldots, m)$ so that a given criterion is optimized.

Another variant of optimal job scheduling, **the jobshop problem**. It is a generalization of the permutation flowshop problem. Formally, referring to Brucker [48], let $M = \{M_1, M_2, \ldots, M_m\}$ and $J = \{J_1, J_2, \ldots, J_n\}$ be two finite sets where the M_i are machines and the J_j are jobs. Let $O = M \times J$, denotes the set of all sequential assignments of jobs to machines, such that every job is done by every machine exactly once; elements $x \in O$ may be written as $n \times m$ matrices.

$$C_{ij} : M \times J \to [0, +\infty[\tag{5.2}$$

Let C_{ij} be the function representing the cost / time for machine M_i to do job J_j. The jobshop problem is to find an assignment of jobs $x \in O$ such that $C(x)$ is a minimum.

5.1.3 Traveling salesman problem

One of the most well-known permutation problems is the traveling salesman problem(TSP). It was first formulated by the Irish mathematician Sir William Rowan Hamilton and the British mathematician Thomas Penynington Kirkman in the 19^{th} century in [35]. Given a list of cities and the distances between each pair of cities, the question stated in such a problem is:

> what is the shortest possible path that visits each city exactly once and returns to the original city? This turns back to calculate the shortest Hamiltonian cycle

There are a wide variety of everyday applications to TSP problems, as they belong to the class of routing problems, such as in the warehouse industry, transportation, supply chain industry, material design, manufacturing, etc.

TSP's permutation representation is seen as a sequence of cities placed in a permutation π of size n, in the order in which they are visited in the tour. And since it is a tour, the first city (the first entry of the permutation) is connected to the last. The overall cost of a feasible tour is the sum of the distances between each pair of adjacent cities of the circuit, and often, it should be minimized, see Eq. 5.3

$$\textit{Minimize} \sum_{i=0}^{n-2} c_{\pi(i)\pi(i+1)} + c_{\pi(n-1)\pi(0)} \tag{5.3}$$

The TSP is a well-studied problem in the literature because of its harshness and the number of its applications. It sounds simple enough, but it is quite difficult to solve such problems because they are NP-hard problems. Starting with von Neumann in 1951 [219], solving TSP problems has motivated the development of important optimization methods including Cutting Planes [74], Branch-and-Bound [32, 113], Local Search [71], Lagrangian Relaxation [96], Simulated Annealing [120].

5.2 Combinatorial Optimization methods

Combinatorial optimization is an optimization subdomain related to operations research, algorithm theory, and computational complexity theory, as well as a combinatorial subdomain related to discrete mathematics. It has important applications in artificial intelligence, auction theory, software engineering, applied mathematics, and theoretical computer science.

Combinatorial optimization is a subject that consists in finding an optimal object from a finite set of objects [191]. Typical problems using combinatorial optimization methods are the TSP, our study problem.

Given the importance of these problems, justified by their great difficulty and by the large number of applications that can be formulated in the form of a combinatorial optimization problem, many resolution methods have been developed. These methods can be roughly classified into two categories: exact (complete) methods which guarantee the completeness of the resolution and approximate (incomplete) methods which lose their exhaustiveness to gain efficiency.

5.2.1 Exact methods

The exact methods can provide an optimal solution, which minimizes or maximizes the value of the objective function. They help to find optimal solutions especially for reasonably sized problems but they generally have difficulties with large applications.

Among the exact methods, the most traditional ones are the Branch and Bound techniques (B & B), initially proposed by Land and Doig [135], or the cutting plane algorithm (the Branch and Cut), initially proposed by Gomory [106].

It is not always possible or reasonable to apply exact resolution methods, due to the complexity of a combinatorial optimization problem (NP-hard problem), and the limited time available to find a solution. Thus, it is useful to use heuristic methods given the possibility of implementing exact methods, the availability of the computation time and the size of the instances to be resolved etc.

5.2.2 Approximate methods

As a general definition, an approximate search method aims to find good quality or near-optimal solution to the problem addressed within a reasonable timeframe by exploring a selected part of the solution space. Good quality solutions are expected but there is no guarantee of optimality of the solutions found.

The most popular search algorithms within the class of heuristic methods are metaheuristics. They are versatile optimization methods that can be applied to any type of problem because they do not contain any specific knowledge of a problem in their design line. It provides a good sufficient

solution to an optimization problem, even with incomplete or imperfect information or limited computational capacity. Thanks to these metaheuristics, approximate solutions can be proposed for classical optimization problems of larger size and for very many requests that were previously impossible to process.

Interest in metaheuristics has grown steadily in recent years, in operations research and artificial intelligence [28, 162, 204]. They are represented mainly by neighborhood-based methods like Simulated Annealing (SA), Tabu Search and Evolutionary Algorithm (EA). This thesis focuses on EAs and more precisely the genetic algorithms (GA).

Evolutionary algorithms are mainly based on the evolutionary theory introduced by Charles Darwin. They are powerful stochastic search and optimization techniques to solve problems that cannot be easily solved in polynomial time and would take far too long to exhaustively process.

As the process of natural selection, there are four main processes in evolutionary algorithms. First, in the process of initialization, an initial population of solutions is created (often randomly) containing an arbitrary number of possible solutions to the problem.

Then, once a population is created, each solution must be evaluated according to a fitness function. Fitness values can be used to calculate the average aptitude of solutions or rank solutions within the population in order to select a portion of the highest rated individuals.

After selecting some solutions, the third process is the generation of a new population, and it may include two sub-process: crossover and mutation. Using the characteristics of the selected parents, new offspring are created with the crossover and mutation operators. These offspring are a mixture of the qualities of the parents.

Finally, there is the termination phase. Either because the algorithm has reached a maximum execution time, or because the algorithm has reached a certain performance threshold. At this point, a final solution is selected and returned. Yu and Gen [231] provided a more recent discussion about evolutionary algorithms.

Fig. 5.1 shows the different steps of the evolutionary algorithm where, after initialization, the population is evaluated then a new population is generated if none of the stopping criteria is met. The process is repeated until a stop criterion is met. A stop criterion can be static like a fixed number of iterations or dynamic, for example, when there are no improvements in fitness values for a certain number of iterations. We can have several stopping criteria in some cases.

Evolutionary algorithms include evolutionary strategies (ES), evolutionary programming (EP), genetic algorithms (GA) and genetic programming (GP). In the following, the interest is in genetic algorithms (GA).

5.3 Genetic Algorithm

Genetic algorithms are one of the most studied classes of EAs. They were developed by John Holland and his collaborators since 1975 [115, 116]. They are powerful stochastic search and optimization techniques for complex problems in a model or abstraction of biological evolution based on Charles Darwin's theory of natural selection. These algorithms have been used to solve

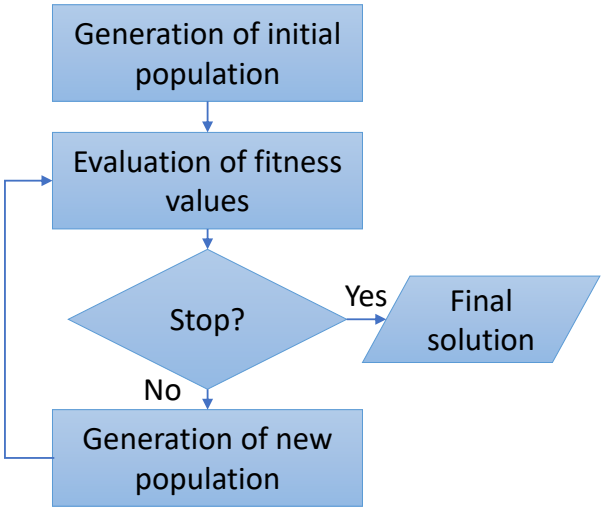


Figure 5.1: Flowchart of an evolutionary algorithm

various kinds of problems [85, 105] and many theoretical and experimental studies have been performed [7, 58, 62, 104, 119, 127, 227].

Fig. 5.2 shows the different steps of GAs through generation to find better solutions using techniques inspired by natural evolution such as inheritance, mutation, selection. As shown in the diagram, in the first step, an initial population of initial individuals is generated. Each individual is represented as a permutation or binary representation or another encoding. By analogy to the evolution theory in biology, an individual is composed of one or several chromosomes. In each chromosome, the genes are the variables of the problem (decision variables) and the possible values are referred to as alleles, see fig. 5.3.

Algorithm 5.1 presents a pseudo-code of a GA.

Generate initial population: $P(0)$
$t \leftarrow 0$
while *stop condition not reached* **do**
  Evaluate $P(t)$
  $P'(t) \leftarrow Selection(P(t))$
  $P'(t) \leftarrow Recombination(P'(t))$
  $P(t+1) \leftarrow Replacement(P'(t), P(t))$
  $t \leftarrow t+1$
end
Return best solution(s)

Algorithm 5.1: Template of a genetic algorithm

During evolution, according to the Darwinian principle of the survival of the fittest, an evaluation function (objective function) is used to associate a "fitness" measurement with each solution indicating its quality. Then the most suitable individuals are selected for reproduction according to

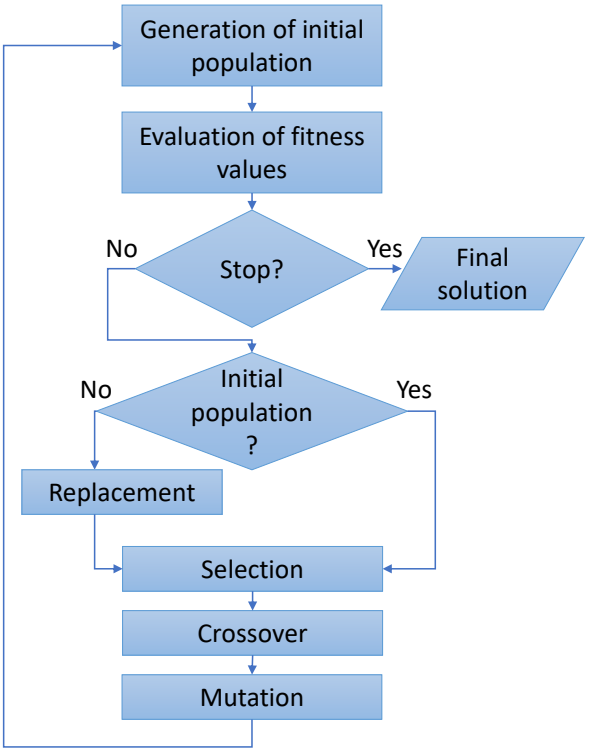


Figure 5.2: Flowchart of genetic algorithm

different selection strategies (Roulette Wheel Selection, Rank Selection, Steady State Selection, Tournament Selection, ...). Then, crossover and mutation, the reproduction operators, are applied to each couple of individuals selected to generate new individuals (offspring). More detailed discussions can be found in the more classic reference by Holland [116], Goldberg [104] and the more recent reference by Gen and Cheng [102], and Gen et al. [103].

Genetic algorithms, over traditional optimization algorithms, have the ability to deal with complex problems and parallelism. They can deal with various types of optimization, whether the objective function is stationary or non-stationary (changes with time), linear or nonlinear, continuous or discontinuous, or with random noise. However, the choice of parameters must be made with care, such as the choice of population size or the mutation and crossover operators. Any inappropriate choice will make it difficult for the algorithm to converge or simply produce meaningless results.

Many approaches can be implemented in the realization of a genetic algorithm (GA). In this thesis and our experiments, the approach that was used is known as the generational approach. In generational GA, a large part of the population is selected and crossed, the resulting offspring are mutated and inserted into the population, thus replacing the old individuals. This means that a temporal or intermediate population is used. Within generation n, it is initialized with a fixed number of individuals from the previous population, and the offspring are inserted into it until the maximum size is reached. At this stage, this temporal population must be in the new population (at generation $n + 1$), a new temporary population is created and the process begins again. Other strategies are presented in the literature (see Goldberg's 1989 book [104] for an introduction) and for comparison between the generational GA and other approaches [73, 160].

Each of the basic components of a GA is detailed below.

Encoding Using the evolutionary algorithm in general and the genetic algorithm in particular to solve optimization problems requires an important task, that is to determine how the solution can be represented according to the elements or terminology of the specific algorithm.

In a population, an individual is characterized by a set of parameters (decision variables) called genes. The genes come together in a chain to form a chromosome (solution) (Fig5.3). The set of genes of an individual is represented in terms of alphabet or numbers.

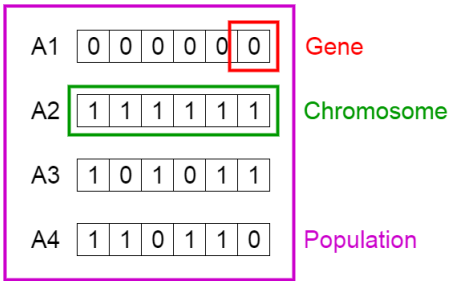


Figure 5.3: Population, Chromosomes and Genes

The processes of generating solutions can produce unfeasible solutions, like solutions with duplicates. It is very important to choose a solution representation that is more likely to produce feasible solutions. The choice of representation can be direct or indirect but the main design consideration is to ensure that each individual generated can still be decoded into a workable solution. For a complex problem, indirect representation is often used with a decoding procedure to convert the indirect representation and to calculate the fitness function for evaluation. In section 5.4, indirect encodings are presented.

Initialization of initial populations It should be emphasized that this step is not so obvious since the initial generation can affect the convergence and quality of the solution in the GA. The importance of the degree of quality and diversification of the initial population is proven and shown in several works [64, 143, 158, 186]. Indeed, the quality of the starting solutions can help to obtain the fittest individuals faster.

Heuristics can be used as a sort of initialization strategy to obtain optimized initial populations [64], taking into account the quality of the solutions, their distribution in the solution space and the cost of calculating the method used to generate this population. For example, part of the population could be generated randomly in order to introduce some diversity while the rest of the population is optimized.

Selection operators Selection mechanisms determine which individuals in the population will be used to produce the next generation according to the fittest parents as they are more likely to generate good quality offspring.

In the literature, a variety of selection methods are proposed. Some of them are based on the fitness of the individuals [67, 220], others are based both on the fitness of the individuals and on some stochastic mechanisms [190]. Roulette wheel selection, also known as Fitness proportionate selection, is a selection operator used in genetic algorithms for selecting potentially useful solutions

for the recombination, see [140] for more details. In roulette wheel selection, the individuals are given a probability of being selected that is directly proportionate to their fitness. Other examples of selection techniques: Binary Tournament, Best Solution Selection, Random Selection, Worst Solution Selection,...

Crossover operators After the selection, recombination stage, using crossover operators, is the most significant phase in a genetic algorithm since it generates new generations. It is the computer transposition of the mechanism which allows, in nature, the production of chromosomes that partly inherit the characteristics of the parents. The crossover operator, as well as the mutation operator, must be designed according to the properties of the problem in question.

The crossover operator is usually a binary operator that combines two selected parents in order to produce better offspring. There are several crossover techniques: Single Point Crossover, Two Points Crossover, PMX Crossover,... In the literature, researchers tend to develop and update existing operator [5, 94] or to create new variation Crossover operator to improve permutation-based Genetic Algorithm [6, 15, 134] See [179, 197] for studies on crossover operators. In section 5.4.1, some specific operators are presented.

Mutation operators After crossing, and for the sake of diversity, the strings are subjected to a mutation since it prevents the algorithm from being trapped in a local minimum. The mutation phase plays the role of recovering lost genetic materials as well as randomly disrupting genetic information.

The mutation operator has traditionally been seen as a simple search operator. It introduces new genetic structures in the population by randomly modifying some of its constituent elements. Mutation operators, such as Uniform Mutation, Swap Mutation, Polynomial Mutation, etc, are widely studied in the literature, see for instance [75, 196, 206].

5.4 Encoding for permutation problems

The use of genetic algorithms allows defining the genotype in the optimization process with different encodings. It refers to the way decision variables are represented. One can use a traditional direct encoding (which is also referred to permutations) or an indirect representation of decision variables (indirect coding) can be used. The choice of the encoding must take into account the problem and the operators to be used (crossover and mutation). This mapping between the set of permutations and the set of their encoding could be used to translate any problem represented by permutations into an equivalent problem represented by another code. This can simplify some of these problems.

The classical permutation allows an easy representation of solutions. However, it can generate duplicates during the crossover and mutation steps. To deal with this issue, specific operators are defined in the literature. For instance, we have PMX, OX, UX for crossover, and Swap, Scramble, and Inversion for the mutation [111]. Quite often, these operators repair to non-viable solutions (those with duplicates). To avoid these repair phases, it is possible to use indirect encodings which are not sensitive to duplicates. In the following, two indirect encoding methods are presented: the Lehmer code and the Inversion table.

Lehmer code Lehmer code, attributed to Derrick Lehmer [137], associates a unique code $L(\pi)$ to each permutation $\pi = \pi_1\pi_2 \dots \pi_n$. $L(\pi)$ is a sequence $L(\pi) = l_1 l_2 \dots l_n$ with l_i the number of elements that are smaller than π_i and that appear to the right of π_i in the permutation.

$$l_i = Card\{j | j > i \quad \& \quad \pi_j < \pi_i\} \tag{5.4}$$

For example, the Lehmer code of the permutation `"5 2 1 4 3"` is `"4 1 0 1 0"`. The elements l_i in the Lehmer code satisfies the condition $0 \leq l_i \leq n - i, \forall i$. The complete process for Permutation to Lehmer code is provided in algorithm 5.2.

Input: Permutation
Output: LehmerCode
l =length of Permutation
for $i = 0;\ i < l;\ i++$ **do**
  counter← 0
  for $j = i + 1;\ j < l;\ j++$ **do**
    if *Permutation[i]* > *Permutation[j]* **then**
      counter←counter+1
    end
  end
  LehmerCode[i] ← counter
end

Algorithm 5.2: Permutation to Lehmer code

Inversion table This type of representation is very similar to the Lehmer Code [125]. The inversion table of a permutation $\pi = \pi_1\pi_2 \dots \pi_n$ is $T(\pi) = t_1 t_2 \dots t_n$ with t_i the number of elements that are greater than i and appearing to the left of i in the permutation.

$$t_i = Card\{j | i < j \quad \& \quad j < \pi_i\} \tag{5.5}$$

For example the inversion table of the permutation `"5 2 1 4 3"` is `"2 1 2 1 0"`. Here $t_3 = 2$ because there are two elements (5 and 2) that are greater than 1 and placed at lower positions in the permutation (to the left of 1). By definition, t_i in the Inversion table always satisfies the condition $0 \leq t_i \leq n - i, \forall i$. The complete process for Permutation to Inversion Table is provided in algorithm 5.3.

5.4.1 Crossover operators

As defined previously in section 5.3, the crossover operator allows generating new solutions from one generation to another. Two Individuals are randomly selected from the population to cross and produce new offspring. Bellow, crossover operators adapted for permutation problems are presented and described.

Single point crossover method consists of choosing a random crossover point and the tails of its two parents are swapped. This results in two offspring, each carrying some genetic information from both parents, see Fig. 5.4.

Input: Permutation
Output: InversionTable
l =length of Permutation
for $i = 0;\ i < l;\ i++$ **do**
    counter← 0
    for $j = 0;\ j < i;\ j++$ **do**
        if *Permutation[i]* $<$ *Permutation[j]* **then**
            counter←counter+1
        end
    end
    InversionTable[i] ← counter
end

Algorithm 5.3: Permutation to Inversion Table

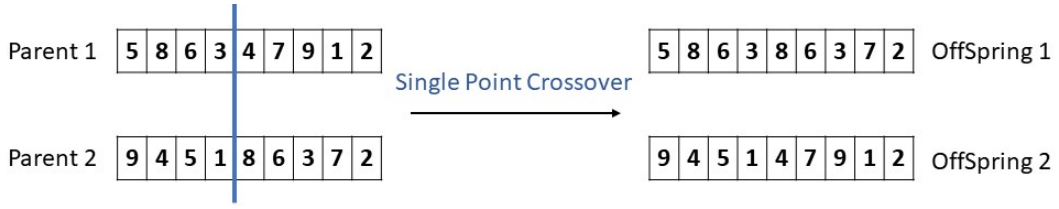


Figure 5.4: Single point crossover

In **two-point crossover**, two crossover points are picked randomly and the gene's value in between the two points are swapped between the parent organisms, see Fig. 5.5. Two-point crossover is equivalent to making two single-point crossings with different crossover points.

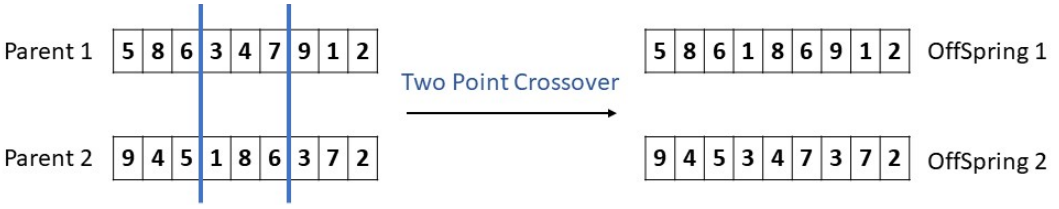


Figure 5.5: Multi-point crossover

Multi-point crossover is a generalization of one-point crossover in which alternate segments are exchanged to obtain new offspring.

The complete process for the k-points crossover method is provided in Algorithm 5.4.

```
Input: Parent1,Parent2,k
Output: Offspring1,Offspring2
l =length of Parent1;
CrossPoints: list of length k;
for i = 0; i < k; i++ do
    do
    p←Math.random(0, l − 1);
    While CrossPoints.contains(p)
    CrossPoints.add(p);
end
Sort CrossPoints;
index← 0;
Boolean exist← false;
for i = 0; i < k; i++ do
    p← 0
    if index < k && i => CrossPoints.get(index) then
        exist=!exist;
        index++;
    end
    if exist then
        Offspring1[i]←parent2[i];
        Offspring2[i]←parent1[i];
    else
        Offspring1[i]←parent1[i];
        Offspring2[i]←parent2[i];

    end
end
```

Algorithm 5.4: k-points crossover

The **partially mapped crossover** (PMX), first introduced by Goldberg and Lingle in [105] is one of the most popular and efficient crossovers for GAs to deal with permutation problems [136].

PMX applies a two-point crossover but additionally uses a mapping relationship to legalize offspring that have duplicate numbers. That means, after choosing two random cutoff points on the parents to build an offspring, the part between the cutoff points, one parent's chain is mapped to the other parent's chain, and the remaining information is exchanged.

For example, in Fig. 5.6, a cutting point is chosen for both parents randomly between the 3^{rd} and 4^{th} gene and another one between the 6^{th} and 7^{th} gene. The substrings between the cut points are called the mapping sections. In our example they define the mappings $1 \longleftrightarrow 8$, $9 \longleftrightarrow 5$, $8 \longleftrightarrow 7$ and $6 \longleftrightarrow 3$. Now the mapping section of the first parent is copied into the second offspring, and the one of the second parent is copied into the first offspring. Then offspring 1 / 2 is filled up by copying the elements of the parent 2 / 1. In case a gene is already present in the offspring it is replaced according to the mappings. For example, the first element of offspring 1 would be a 1 like the first element of the first parent. However, there is already a 1 present in this offspring. Hence, because of the mapping $1 \longleftrightarrow 8$, we choose the first element of offspring 1 to be 8. But since also 8 is already in offspring 1, because of the mapping $8 \longleftrightarrow 7$, we choose the first element of offspring 1 to be 7. Similarly for the rest and for the second offspring as well.

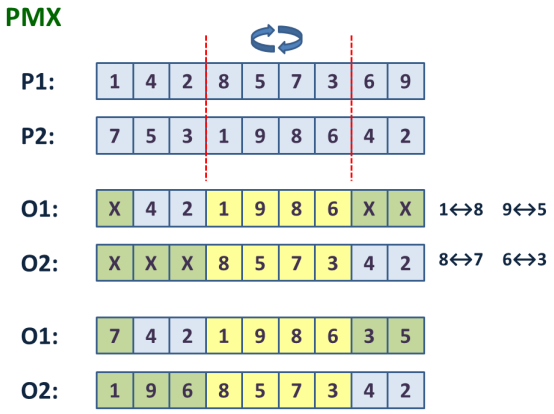


Figure 5.6: PMX crossover example

5.5 Performance indicators

It is necessary to use performance indicators to measure how each kind of operator keeps some characteristics of any considered permutation problem.

Hamming distance The Hamming distance is a known mathematical distance metric. It was initially defined to measure how two binary strings differ from each other and then it has been generalized for permutations. It is an indicator that can be used to assess diversity. It measures the similarity between two solutions concerning the decision variables. Considering two permutations say π and π', it is equal to the number of positions in which π differs from π' (Eq. 5.7).

$$D(\pi, \pi') = \sum_{i=0}^{n-1} x_i \tag{5.6}$$

where

$$x_i = \begin{cases} 1 & if \quad \pi_i \neq \pi'_i \\ 0 & Otherwise \end{cases} \tag{5.7}$$

Note that low Hamming distances mean that two permutations (π, π') are almost identical

Now, there are different indicators to assess how different genetic properties are transmitted from parents to offspring.

Edge Based Indicator (EBI) The Edge Based Indicator [81] permits to observe the transmission of properties from parents to offspring. It is more particularly related to edges. It counts the edges which are present both in parent and in offspring, using Eq. 5.8, which is reminded just below to make the reading easier. This metric must be maximized.

$$EBI = \frac{NET \times 100}{2n} \tag{5.8}$$

Where the total number of edges (NET) is:

$$NET = \sum_{i=1}^{n} \sum_{j \neq i}^{n} NE(v_i, v_j) \tag{5.9}$$

Where the number of edges (NE) is:

$$NE(v_1, v_2) = \begin{cases} 2 \quad if \begin{cases} & v_1 \text{ immediately proceeds } v_2 \text{ in Parent 1} \\ and & v_1 \text{ immediately proceeds } v_2 \text{ in Parent 2} \\ and & v_1 \text{ immediately proceeds } v_2 \text{ in the offspring} \end{cases} \\ 1 \quad if \begin{cases} & v_1 \text{ immediately proceeds } v_2 \text{ in Parent 1} \\ or & v_1 \text{ immediately proceeds } v_2 \text{ in Parent 2} \\ and & v_1 \text{ immediately proceeds } v_2 \text{ in the offspring} \end{cases} \\ 0 \quad Otherwise \end{cases} \tag{5.10}$$

Position based indicator This indicator more specifically deals with positions. It is defined by Eq. 5.5.

$$PBI = \frac{\sum_{i=1}^{2} \sum_{j=1}^{2} PB_{ij}}{4} \times \frac{6}{n(n+1)(2n+1)} \tag{5.11}$$

where PB_{ij} is the euclidean distance

$$PB_{ij} = \sqrt{\sum_{k=1}^{n} PO_j(k) - PP_i(k)} \tag{5.12}$$

where

$PP_i(k)$ is the position of k in the parent i, $i \in \{1, 2\}$,

and

$PO_j(k)$ is the position of k in the Offspring j, $j \in \{1, 2\}$. This indicator must be minimized

5.6 Conclusion

This chapter gave a general overview of optimization problems based on permutations as an excellent example of the direct application of permutations and their properties. Combinatorial optimization methods are divided into two large families: exact methods and heuristic methods. In this thesis, we are interested in genetic algorithms, powerful heuristic stochastic search and optimization techniques for complex problems, in particular permutation-based problems. A detailed explanation of the different stages of the GA is also presented, with the definition of certain operators. In the next two chapters, we will present the contributions made on the optimization of permutation-based problems. More precisely, we will focus in these works on the indirect encodings of decision variables.

Chapter 6

Transmission of Genetic Properties in Permutation Problems

6.1 Introduction

Permutation-based optimization problems are widely studied in the literature because of their hardness and the diversity of their application fields. They are used in different domains, for instance, the network devices deployment, scheduling or transportation. Solving such problems consists in finding a permutation that minimizes / maximizes some criteria.

When dealing with permutation problems, one should ensure that there are no duplicates in the permutation. Using a genetic algorithm, this can be easily verified when generating the initial solutions. However, over generations, genetic operators such as crossover and mutation can duplicate alleles (values). There are mainly two approaches to avoid these repetitions. The first is to use mutation and crossover operators which repair individuals containing duplicates (e.g. OX, UX, or PMX [111]). Another method consists in using a solution encoding (indirect encoding) which tolerates, duplicates and defines a bijection between this indirect encoding and the permutation. The Lehmer code and the Inversion table are two examples of indirect encoding. Also, two non-classical encodings are introduced in this chapter. However, it should be noted that part of parents' genetic properties can be lost when generating offspring if indirect encoding is used [155]. This chapter studies the transmission of genetic properties focusing on both decision variables and objective function domains. We consider a single objective traveling salesman problem (TSP) as an example of a permutation optimization problem.

This chapter is organized as follows. A brief overview of related works is introduced in Section 6.2. Section 6.3 describes classical direct and indirect encodings, and two non-classical new indirect encodings, as well as their associated operators. Section 6.4 presents our experiments and results regarding the transmissions from parents to offspring and the fitness improvement for three different problems. Finally, conclusions are given in Section 6.5.

6.2 Related work

Initially proposed by Holland and Goldberg, genetic algorithms (GAs) have quickly evolved to solve multi-objective problems. Since Holland introduced the main mechanisms of GAs [115],

many authors explained that choosing the most suitable representation / encoding is one of the major issues [31, 68, 81, 104, 110, 141, 155, 164, 214]. Among these authors, Goldberg [104] studied the behavior of representations and crossover-based GAs. He explained, in the schemata principle, how this representation - crossover combination should allow the transmission of meaningful building blocks from parents to offspring. Recently, Mohammed Ali et al. [155] show the impact of the choice of the couple encoding - crossover operator on the resolution of the permutation problems. They compare the characteristics related to the transmission of the properties between the parents and the offspring.

Djerid et al. [81] considered permutation encoding and crossover in a way similar to the classical schema theory [104]. They explain that encoding and crossover should be adapted according to which properties of the parents should be inherited by offspring. Pesko [164] presents an evolution algorithm for solving small (up to 32 nodes) constrained TSP. This new differential evolution algorithm with only two parameters (the population size and the number of the generations) uses the Lehmer code to encode solutions. In [214], Ücoluk proposes an inversion sequence as the representation of a permutation. This method is used for solving TSP and is compared to the well-known PMX crossover method. It is observed that Ücoluk's method outperforms PMX in convergence rate by a factor which can be as high as 11.1, on a cost of obtaining slightly worse solutions on average. Bekiroğlu [31] uses new alternatives of encoding types such as Quaternary encoding and octal encoding. He examined how they contribute to the efficiency and robustness of the genetic algorithm. He concluded that it is not possible to claim that one of the encoding types is exactly dominant over the others in all aspects such as convergence, finding the optimum solution, and the number of iteration. In [110], a GA is proposed to optimize the weight of steel truss structures. The obtained results proved the effectiveness of the genetic algorithm in relation to the classical genetic algorithm. In this case, the set of design variables consists of the collection of profiles manufactured in steel mills. Obviously, this set of profiles is discrete. The most effective type of encoding in such a case is value encoding.

In [188], Rosa et al. discuss the teachers' placement (in elementary school) problem based on genetic algorithms by finding a chromosome that represents the possibility of teachers placement solution, composing a population, and finding the recommended combination of two selected mutations operators and two selected crossover operators to achieve optimal results. Another work that studies the choice in selection, crossover, and mutation operators and their impact on the performance of a genetic algorithm is [161]. The authors present a novel framework for an adaptive and modular genetic algorithm (AMGA) to discover the optimal combination of the operators in each stage of the GA to avoid premature convergence.

6.3 Encoding and recombination operators

Encoding refers to the way decision variables are represented. When using genetic algorithms, it allows to define the genotype in the optimization process. One can use a direct encoding or an indirect representation of the decision variables (indirect encoding). The choice of the encoding should take into consideration the problem and the operators to use (crossover and mutation). This mapping between the set of permutations and the set of their encoding could be used to translate any solution represented by permutations into an equivalent solution represented by another code. This may simplify some problems. Bellow, we will present the direct encoding, permutation, as well as two classical indirect encodings, Lehmer code and Inversion table, and finally we will make an introduction to two non-classical encodings, Transposition array and Inverse Transposition array.

6.3.1 Direct Encoding

A permutation $\pi = \pi_1\pi_2 \dots \pi_n$ is an arrangement of the numbers $1, 2, \dots, n$ for some positive n. Each permutation is represented by a unique number that represents its order among the $n!$ possible permutations enumerated following a lexicographic order. The classical permutation allows an easy representation of solutions. However, it can generate duplicates during the crossover and mutation steps. To deal with this issue, specific operators are defined in the literature. We can list PMX, OX, UX for the crossover, and Swap, Scramble, and Inversion for the mutation [111]. Quite often, these operators repair to non-viable solutions (those with duplicates). To avoid these repair phases, it is possible to use indirect encodings which are not sensitive to duplicates. In the following, we define four indirect encoding methods, some are classical, and others are new.

6.3.2 Classical Encodings

As a classical encoding, we can cite several ones, but in this chapter, we are interested in the **Lehmer code** and the **Transposition table**. These encodings are described in detail in the section 5.4. These are considered classic because there are many studies about then. For example, the article by Mohammed Ali et al. [156] deals with evolutionary algorithms and studies the behavior of an evolutionary design, based on a Lehmer code representation coupled to a simple k-point crossover. Moreover, Mehdi studied in her thesis [153] permutations as natural numbers and proposed an encoding as a new way of representing the solution space of permutation problems in metaheuristics. This coding approach is based on Lehmer codes and inversion tables.

The use of an indirect representation requires coding and decoding operations. Fig. 6.1 and 6.4 illustrate the steps of the process of crossover using Lehmer code and Inversion table. For instance, in Fig. 6.1, the parents are represented first as a classic permutation (Fig. 6.1(a)). Then, they are coded using the Lehmer code (Fig. 6.1(b)) before applying the crossover operator and generating offspring (Fig. 6.1(c)). These offspring are then decoded to be presented as a classical permutation (Fig. 6.1(d)).

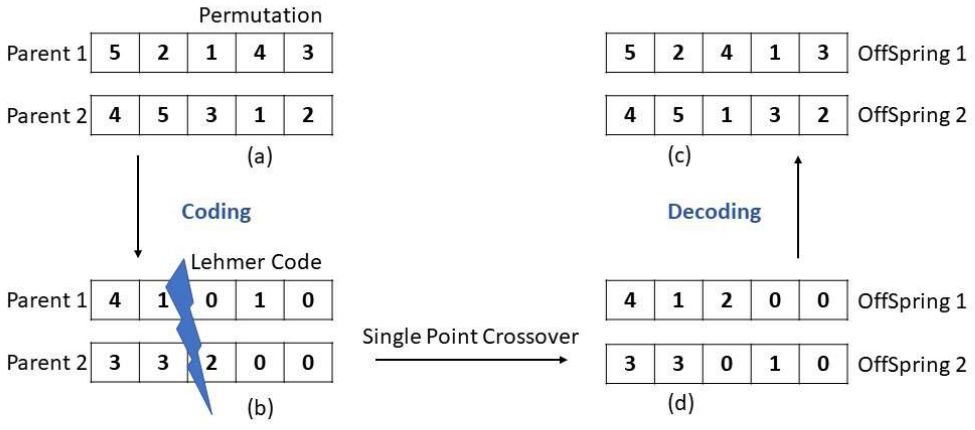


Figure 6.1: Applying crossover on Lehmer code encoding

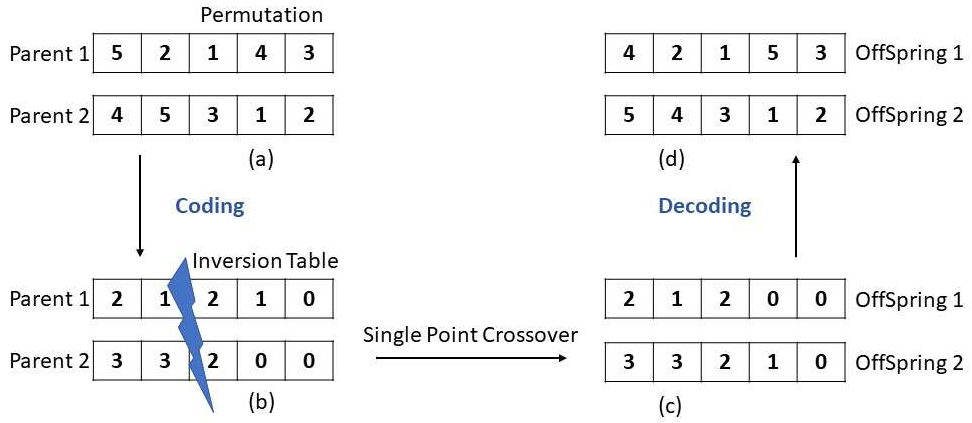


Figure 6.2: Applying crossover on Inversion table encoding

It is important to note here that the offspring illustrated in Fig. 6.1(d) (or Fig. 6.2(d)) have alleles that are not inherited from their parents. For example, for the offspring of Fig. 6.1(d), the sequences "4 1 3" and "1 3 2" are not inherited from any parent. This problem can be rephrased as: the sequences "1 4 3" and "3 1 2" of the parents (Fig. 6.1(a)) are lost during the recombination process (without a mutation having been carried out).

6.3.3 New encodings

Transposition array Inspired by the work of J-L. Baril [19, 20] and the work of Mantaci [151], we are interested in an encoding that, in [19], is used in order to obtain Gray codes for restricted classes of length n permutations. Let $\mathfrak{S}_n$ be the set of permutations of length n. We represent a permutation $\pi \in \mathfrak{S}_n$, where $\pi = \pi_1\pi_2 \dots \pi_n$. Moreover, if $\gamma = \gamma(1)\gamma(2)\dots\gamma(n)$ is a length n permutation then the product $\gamma \cdot \pi$ is the permutation $\gamma(\pi_1)\gamma(\pi_2)\dots\gamma(\pi_n)$. In $\mathfrak{S}_n$, a k-cycle $\pi = \langle i_1, i_2, \dots, i_k\rangle$ is a length n permutation verifying $\pi(i_1) = i_2, \pi(i_2) = i_3, \dots, \pi(i_{k-1}) = i_k$, $\pi(i_k) = i_1$ and $\pi(j) = j$ for $j \in [n]\backslash\{i_1, \dots, i_k\}$. In particular, a 2-cycle is called a transposition. Let $C_n \subset \mathfrak{S}_n$ be the set of n-cycles. The elements of C_n will be called cyclic permutations (or cycles for short). Obviously, C_{n+1} and $\mathfrak{S}_n$ have the same cardinality. Any permutation $\pi \in \mathfrak{S}_n$ is uniquely decomposed as a product of transpositions of the following form (Eq. 6.1):

$$\pi = \langle p_1, 1\rangle \cdot \langle p_2, 2\rangle \cdot \langle p_3, 3\rangle \dots \langle p_n, n\rangle = \prod_{i=1}^{n} \langle p_i, i\rangle, \tag{6.1}$$

where π_i are some integers such that $1 \leq \pi \leq i \leq n$. Conversely, any such decomposition provides a permutation in $\mathfrak{S}_n$. Therefore, Eq. 6.1 yields a bijection from $\mathfrak{S}_n$ to the product set $T_n = [1] \times [2] \times \dots \times [n]$. Then we have another way to represent a permutation:

Definition 16. The transposition array (TA) of a permutation $\pi = \prod_{i=1}^{n} \langle p_i, i\rangle \in \mathfrak{S}_n$ is defined by $p_1p_2 \dots p_n \in T_n$

For example, if $\pi = 145632$ then its decomposition into transpositions is $\langle 1, 1\rangle \cdot \langle 2, 2\rangle \cdot \langle 3, 3\rangle \cdot \langle 2, 4\rangle \cdot \langle 3, 5\rangle \cdot \langle 4, 6\rangle$, and its corresponding transposition array is 123234.

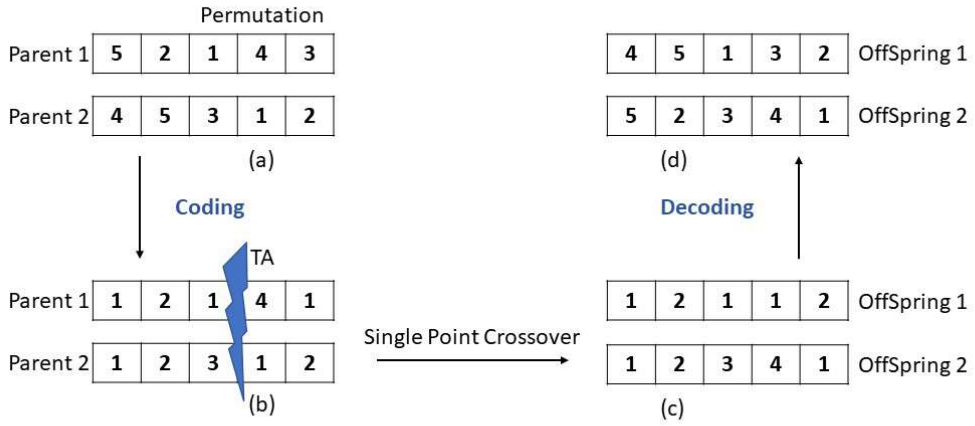


Figure 6.3: Applying crossover on Transposition array encoding

Fig. 6.3 illustrates the steps of the process of crossover using the Transposition array encoding where, at first, the parents are represented as a classic permutation then, they are coded using into TA encoding before applying the crossover operator and generating offspring. Then these offspring are then decoded to be presented as a classical permutation. In this case as well, the offspring illustrated in Fig. 6.3(d) have alleles that are not inherited from their parents. For example, the sequences "3 2" and "4 1" of the offspring are not inherited from any parent.

Inverse transposition array Another decomposition is proposed to any permutation $\pi \in \mathfrak{S}_n$ also as a product of transpositions of the following form:

$$\pi = \langle p_1, n\rangle \cdot \langle p_2, n-1\rangle \cdot \langle p_3, n-2\rangle \ldots \langle p_n, 1\rangle = \prod_{i=1}^{n} \langle p_i, n-i+1\rangle, \tag{6.2}$$

where π_i are some integers such that $1 \leq \pi \leq i \leq n$. Conversely, any such decomposition provides a permutation in $\mathfrak{S}_n$. Therefore, Eq. 6.2 yields a bijection from $\mathfrak{S}_n$ to the product set $T_n = [1] \times [2] \times \cdots \times [n]$. Then we have another way to represent a permutation:

Definition 17. The inverse transposition array (ITA) of a permutation $\pi = \prod_{i=1}^{n} \langle p_i, n-i+1\rangle \in \mathfrak{S}_n$ is defined by $p_1 p_2 \ldots p_n \in T_n$

For example, if $\pi = 4312$ then its decomposition into transpositions is $\langle 2, 4\rangle \cdot \langle 1, 3\rangle \cdot \langle 1, 2\rangle \cdot \langle 1, 1\rangle$, and its corresponding inverse transposition array is 2111.

Fig. 6.4 illustrates the steps of the process of crossover using the Inverse transposition array encoding.

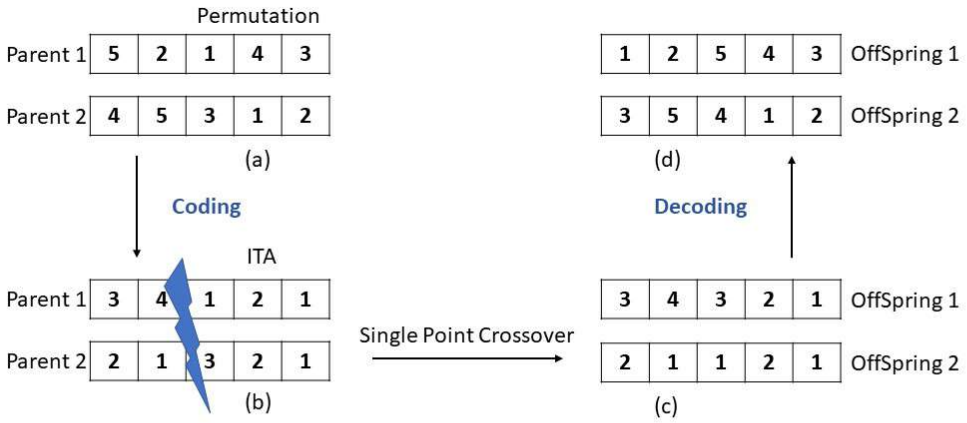


Figure 6.4: Applying crossover on Inverse transposition array encoding

In the case of ITA also, the offspring illustrated in Fig. 6.4(d) have alleles that are not inherited from their parents and were lost during the recombination process (without a mutation having been carried out). It is therefore important to study the impact of the loss of these genetic characteristics over generations.

6.4 Experiments and results

Our goal is, first, to study the transmission of characteristics from parents to offspring. For this, we use metrics that focus on the similarities between solutions (Section 6.4.1). Then, we evaluate the quality of the results of each encoding with respect to the objective function (Section 6.4.2). For this study, we considered three TSP problems proposed in the TSPLIB benchmark [213]: `eil51`, `att48`, and `burma14`. These problems instances are of 51, 48, and 14 cities respectively.

The experimentations carried out during this study use the **jMetal** framework [1]. jMetal stands for metaheuristic algorithms in Java, and it is an object oriented Java framework for single / multi-objective optimization with metaheuristics. The framework's object-oriented architecture and the included features allowed us to experiment with the classic and advanced techniques provided in the state-of-the-art, to develop our own algorithms and to solve optimization permutation problems. For more informations, two papers for Nebro, Durillo and Vergne [92,159] describes jMetal.

The results presented in this section correspond to the averages of 10 independent runs. The experiment parameters used are summarized in Table 6.1.

Encoding	Crossover	Mutation	Population size	Nb. of generations
Permutation	PMX	Swap	200	300
Lehmer code	2-point	BitFlip		
Inversion table	2-point	BitFlip		
Transposition array	2-point	BitFlip		
Inverse transposition array	2-point	BitFlip		

Table 6.1: Experiment parameters

6.4.1 Assessment of transmissions from parents to offspring

The goal is to evaluate the transmission of the genetic characteristics from parents to offspring in order to observe the impact of the different encodings. For this purpose, several indicators are used: the Hamming distance, the Edge based indicator and the Position based indicator.

Hamming Distance

As defined in section 5.5, this metric measures the similarity between two solutions with respect to the decision variables. Each hamming distance (HD) value in this study is the maximum value between the distance of the solution and each of its parents.

Fig. 6.5 and Fig. 6.6 show the performance of the Hamming distance between parents and offspring over generations for the problems `eil51` and `att48`. We observe that this metric gradually decreases over the generations because the genetic algorithm will tend to exploit the neighborhood of the best solutions. However, due to the appearance of new alleles (as shown in Fig. 6.1 and 6.2), the Lehmer code and the Inversion table show higher HD values than the other encodings. But also, the Transposition array and the Inverse transposition array encodings, even if they display lower HD values than the other two indirect encodings, still manage to decrease and remain superior to the direct permutation encoding. This is due to the same previous reason, the appearance of new alleles after recombination (as shown in Fig. 6.3 and 6.4).

Fig. 6.7 shows this performance in the case of the problem `burma14`. We have the same aspect like the other problems but in a condensed way, since the chromosome length is relatively small (14 cities).

For the permutation encoding there may be a lack of diversity for the permutation encoding case since after the 100^th generation, the HD tend to be zero, which means that the parents and the offspring become almost identical. Based on the results of this metric, the diversity of solutions is better using Transposition array and Inverse transposition array, and even better using the Lehmer code, and Inversion table shows the best diversity.

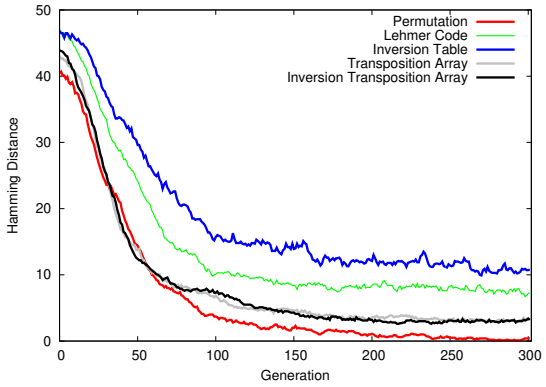


Figure 6.5: Hamming Distance for `eil51`

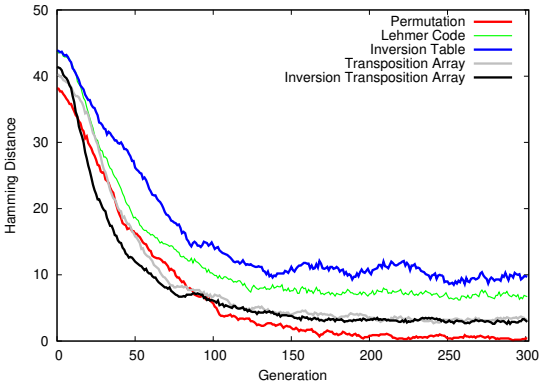


Figure 6.6: Hamming Distance for `att48`

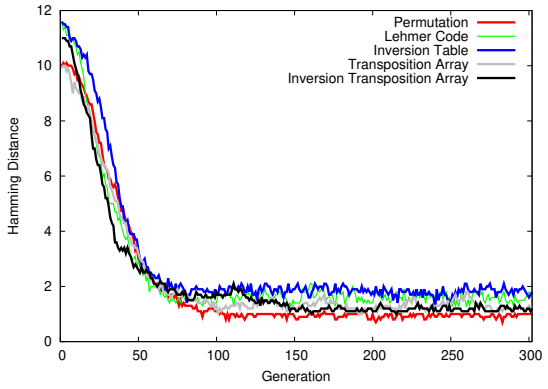


Figure 6.7: Hamming Distance for Bruma14

Edge Based Indicator

The EBI metric counts the edges which are present both in parent and in offspring. It is defined in section 5.5. The greater this value, the better the hereditary transmission occurs between parents and offspring.

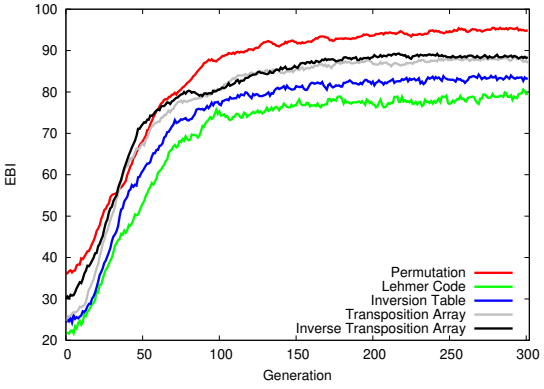


Figure 6.8: EBI for `eil51`

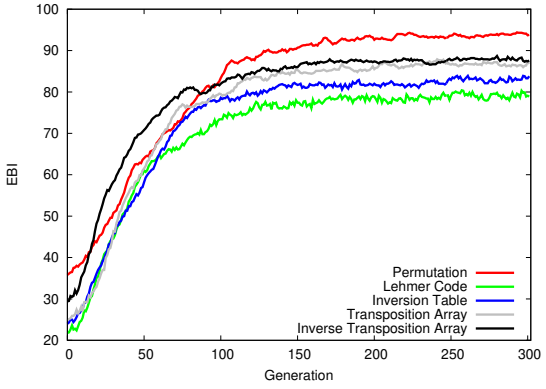


Figure 6.9: EBI for `att48`

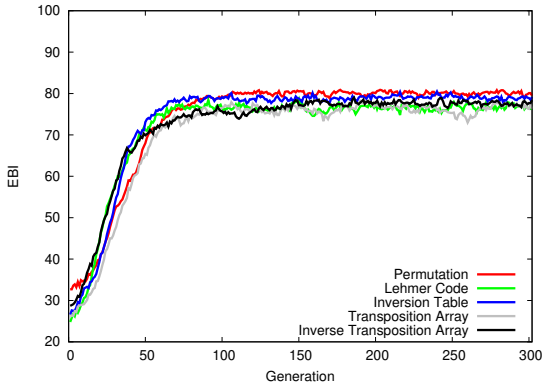


Figure 6.10: EBI for `burma14`

Fig. 6.8 and 6.9 show the variation of EBI values for `eil51` and `att48`. Although not all parent genes are correctly copied in offspring when using the indirect encodings, these four, in addition

to the permutation, achieve high values of EBI. Which means that offspring retain and inherit the edges presented in their parents. For indirect encodings, we tend to have slightly lower values of EBI than the classic permutation, since the red curve increases clearly higher than the rest. This means that the edges are preserved after crossover when using indirect encodings but not in a large quantity that can lead to have almost identical solutions.

In Fig. 6.10, the curves representing the EBI values for the `burma14` problem are superimposed. But if we take a closer look, the red line referring to the permutation encoding shows the highest values. But overall, all representations show good EBI values.

Position Based Indicator

The PBI metric looks at the position of cities among parents and offspring. The weaker this indicator, the more the positions are respected (see section 5.5).

Fig. 6.11 and 6.12 illustrates the variation of the PBI according to the five proposed encodings for `eil51` and `att48`. In terms of the PBI values for the case of permutation, it decreases to reach of the value zero with small perturbations, this means that, at some point, starting from around the generation 100, there is no distance between the permutations representing the parents and the offspring. The curves representing the other four encodings strongly decreases and tends towards zero . This can be explained by the diversity of individuals in these cases, since the distance from the parent is low but not zero.

Fig. 6.13 shows the performance of PBI for `burma14`. The variation is almost the same for all encodings. This can be explained by the relatively small chromosome length (14 cities), which means that the probability of having different positions for cities is low. But we can see that there is a faster decline for the indirect encodings.

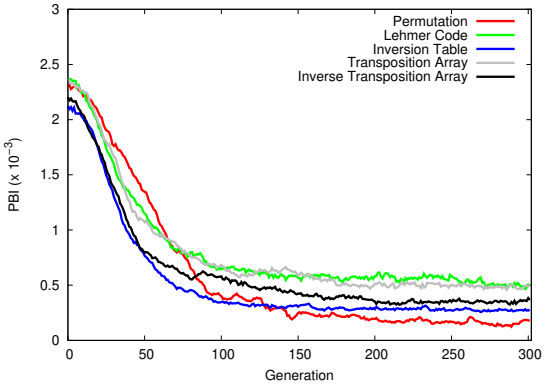


Figure 6.11: PBI for `eil51`

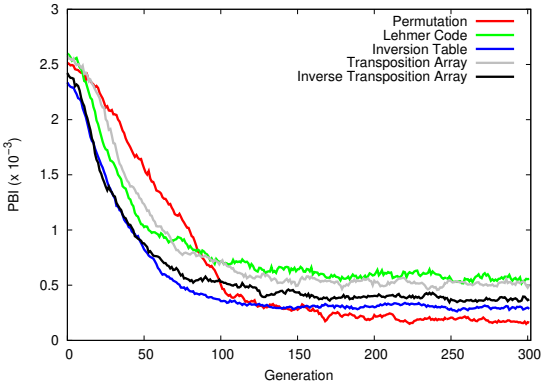


Figure 6.12: PBI for `att48`

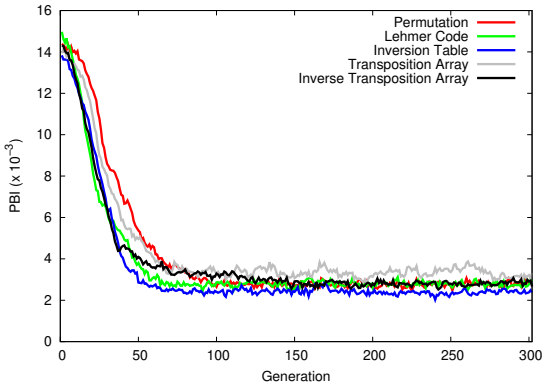


Figure 6.13: PBI for `burma14`

6.4.2 Analysis of fitness distribution

In this section, we study the evolution of the fitness value over generations and compare the performance of the different types of encoding. This is done using a method, inspired by the set of experimentation introduced by Portman and Vignier in [183]. They classify solutions, for each generation, in five groups based on the fitness values. These groups represent five equal intervals of values such that the overall interval is bounded by the minimal and maximal values of the objective function at each generation. This means, at each generation, the intervals that define the groups depend on the values in this specific generation. Solutions for the first group (group 1) are better than the ones while the last group (group 5) is the worst.

At each generation, the solutions will be distributed in each quintile called group 1 to group 5. It is expected that the number of solutions in group 1 increases over the generations. This group would therefore become predominant over the others. However, if almost all the solutions were found in a single group, this could show a premature convergence and a possible trap due to a local optimum.

In summary, the distribution of the population into five quintiles as proposed by Portmann and Vignier [183], makes it possible to analyze the structure of the population. This allows, among other things, to observe the distribution of solutions in the space of objective functions. The distribution being made generation by generation, it is not necessary (even, it could be counterproductive) to have almost all the solutions in Group 1. On the other hand, this group must be majority to illustrate a good structure of the population. In our study, such a distribution would reflect a diversity in the space of the objective functions (which could be supplemented by the values of the indicators such as the Hamming distance, the EBI and the PBI in the space of the decision variables).

`Eil51` Problem

Fig. 6.14 shows that for the classical permutation, after the 100^{th} generation, almost all the solutions are in group 1. The other groups are barely represented. This is in accordance with the Hamming distance of the permutation (see Fig. 6.5) where it was observed that parents and offspring were almost identical after the 100^{th} generation. This situation may cause a lack of diversity.

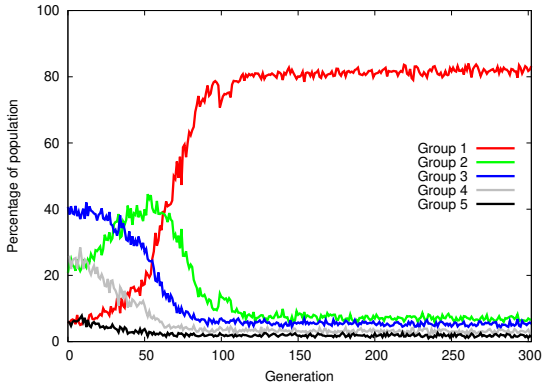


Figure 6.14: Classification using permutation encoding for `eil51`

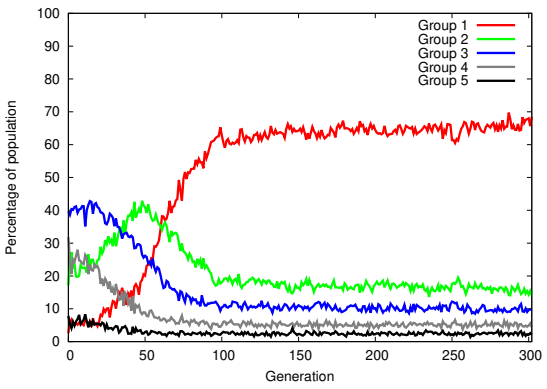


Figure 6.15: Classification using Lehmer Code encoding for `eil51`

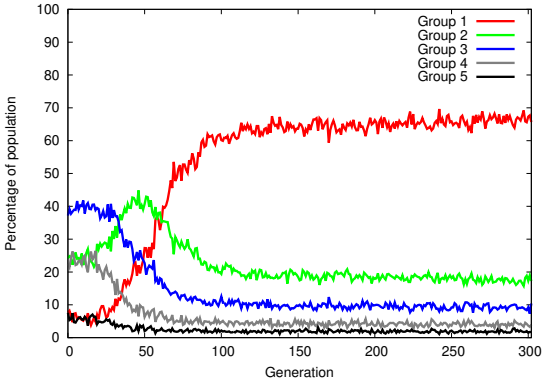


Figure 6.16: Classification using Inversion Table encoding for `eil51`

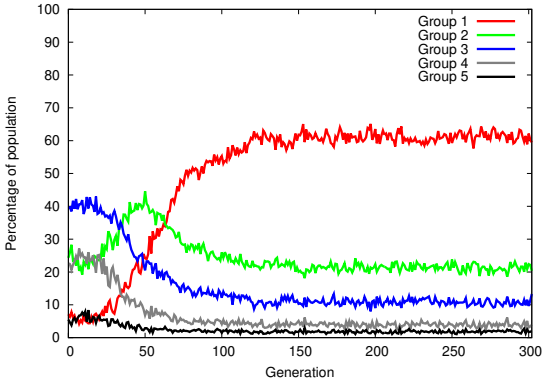


Figure 6.17: Classification using Transposition array encoding for `eil51`

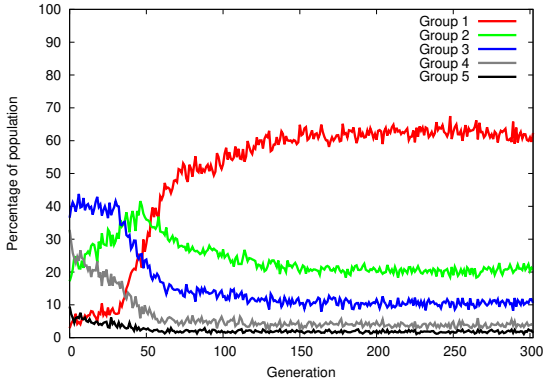


Figure 6.18: Classification using the inverse Transposition array encoding for `eil51`

The rest of the figures representing the classification of individuals for indirect encodings show that solutions of group 1 correspond to the majority, but the other groups proportions of the solutions are not negligible . This is explained as a good diversity. A small difference may hold our attention, for Lehmer code and IT, in Fig. 6.15 and 6.16, the first group reaches 70% while for the TA and ITA, the best group reaches 60% of the population (see Fig. 6.17 and 6.18). But this is not a sign of poor quality, since the percentage of the number of solutions in the top two groups represents

between 70 and 75% for all indirect encodings.

`att48` Problem

For `att48`, we can also see a problem of diversity. In Fig. 6.19 that represents the classification of groups using the classical encoding, almost all the solutions are in group 1 after the 100^{th} generation while the other groups are barely represented.

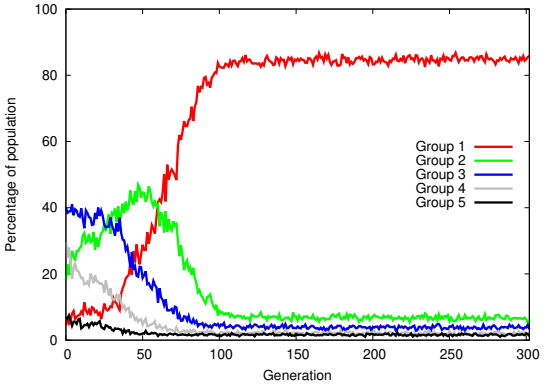


Figure 6.19: Classification using permutation encoding for `att48`

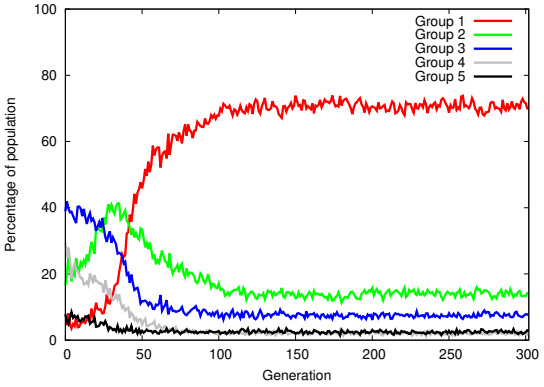


Figure 6.20: Classification using Lehmer Code encoding for `att48`

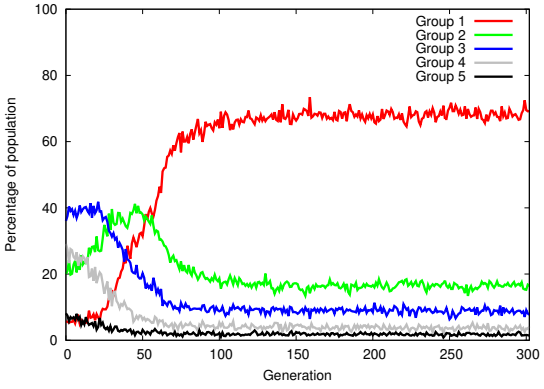


Figure 6.21: Classification using Inversion Table encoding for `att48`

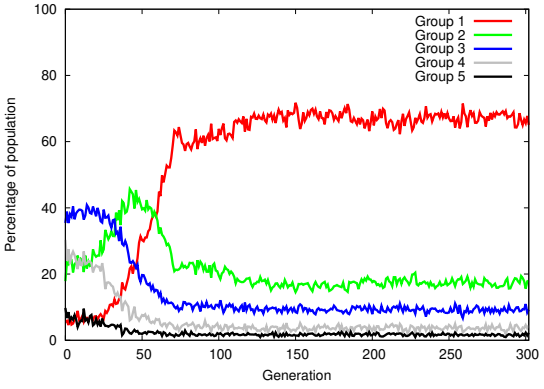


Figure 6.22: Classification using Transposition array encoding for `att48`

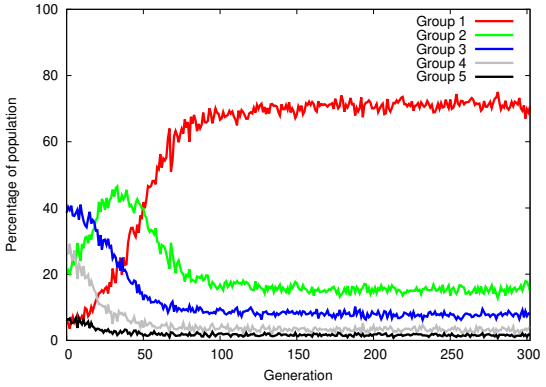


Figure 6.23: Classification using the inverse Transposition array encoding for `att48`

Fig. 6.20, 6.21, 6.22 and Fig. 6.23 show that the indirect encodings present greater diversity since the solutions of group 1 correspond to the majority, but the proportions of the solutions in the other groups are not negligible. This helps alleviate the pressure from elitism.

`Burma14` Problem

The diversity problem is also presented in the case of `burma14` problem. A simple comparison between Fig. 6.24 and the other figures (Fig. 6.25, Fig. 6.26, Fig. 6.27 and Fig. 6.28) shows how groups 3, 4 and 5 are barely represented after the 50^th generation when using permutation encoding and how for the other encodings, these groups are better represented keeping the majority of the individuals in the best group.

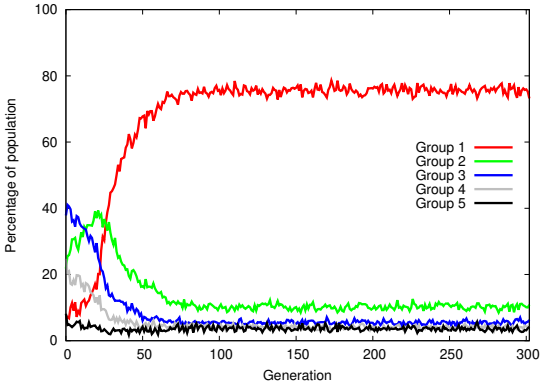


Figure 6.24: Classification using permutation encoding for `burma14`

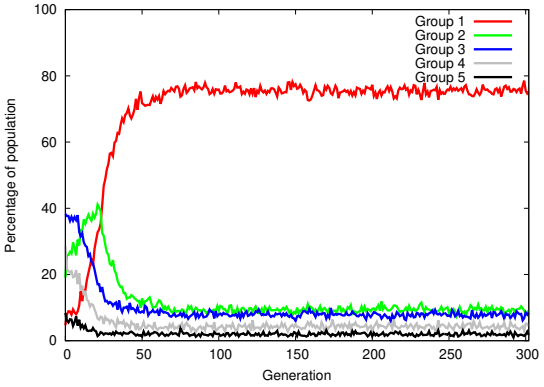


Figure 6.25: Classification using Lehmer Code encoding for `burma14`

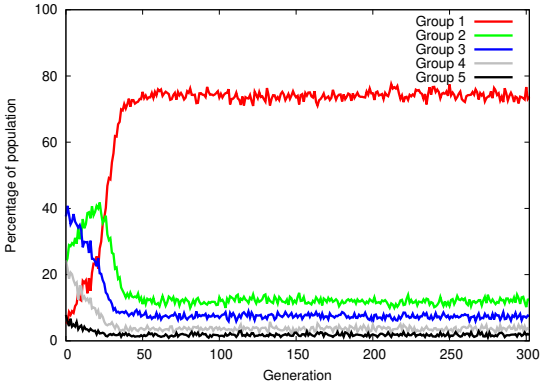


Figure 6.26: Classification using Inversion Table encoding for `burma14`

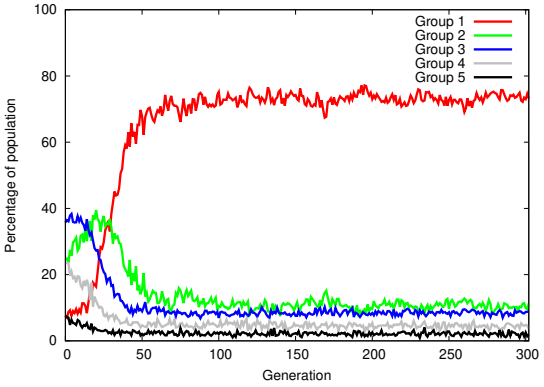


Figure 6.27: Classification using Transposition array encoding for `burma14`

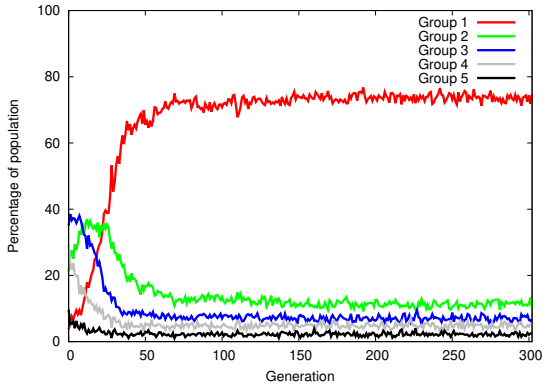


Figure 6.28: Classification using the inverse Transposition array encoding for `burma14`

6.5 Conclusion

This chapter studies the transmission of genetic characters in an optimization process. It presents five encodings. The first, called direct encoding is the classical permutation. The other four are referred to as indirect encodings: the Lehmer code, the Inversion tables, the Transposition array, and the Inverse transposition array. Indirect coding has the advantage of not being sensitive to the appearance of duplicates during crossover operations. However, there is a partial loss of the genetic properties between parents and offspring. The study of the impact of these encodings was made both in the space of decision variables (thanks to the Hamming distance, EBI, PBI) and in the space of fitness values (distribution of solutions in quintiles). The results show that indirect encoding makes it possible to preserve diversity within populations without losing quality in terms of the objective function (the solutions in group 1 are the majority and the population remains diverse).

Chapter 7

Schema Conservation Study in Permutation Problems

7.1 Introduction

The success of an optimization process based on a genetic algorithm relies on the transmission of the genetic heritage between parents and offspring. Relevant information from parents should be passed on to offspring and this goes through a crossover operation. This operator must therefore preserve this good transmission. So to create more efficient genetic algorithms, it is instructive to look at how evolution works from generation to generation.

One of the first attempts to understand how genetic algorithms (GAs) work in a formal sense was the use of schema theory, developed in 1975 by John Henry Holland. He realized that when the genetic algorithm assesses the fitness of a solution in the population, it is in fact assessing the fitness of many solutions in an implicitly parallel mode.

The objective of this chapter is to study the transmission of genetic properties when using indirect encodings. We focus on the Lehmer code (LC) and the Inverse transposition array (ITA) encoding as they preserve the prefix / suffix after the crossover operator. An adaptation of the crossover process is proposed to reinforce the preservation of the schema of good parents. To do this, we will study the conservation of the suffix or prefix for the Lehmer code and the Inverse transposition array.

First, a brief overview of schema theory in genetic algorithms is shown along with some related work. Then a detailed section is devoted to explain schema conservation for LC and ITA encoding. After that, we will present the proposed method to apply the adaptation in question and then the results are presented with descriptions and interpretations. Finally, this chapter ends with a conclusion and final remarks.

7.2 The Schema Theory for Genetic Algorithms

Although "survival of the fittest" has worked relatively well in the real world, the question remains about how the concept of a genetic algorithm actually works in a computer. To explain how this algorithm works, John Holland, introduced the schema theory [116], then it was popularized by Goldberg [47] .

Holland's schema theory is widely regarded as the foundation for explanations of the power of genetic algorithms (GAs) and one of the earliest attempts to understand how these algorithms work in a formal sense. Holland realized that when the genetic algorithm evaluates the fitness of a solution in the population, it is in fact evaluating the fitness of many solutions in an implicitly parallel mode. Specifically, in schema theory, the search space is divided into subspaces of different levels of generality, and mathematical models are built that estimate how the number of individuals in the population belonging to a certain schema can increase over the next generation.

A schema is a set of binary strings that match the template for schema H. More precisely, supposing that we have a binary string, a schema is a string of length l from the alphabet $\{0, 1, *\}$ where the $*$ is a wildcard that represents either a 0 or 1. The defining length of a schema is the distance between the outermost non wildcard symbols. When a string matches the schema on the fixed positions it is called an instance of the schema.

Through the process of reproduction between individuals, the most fit schemata[1] will increase or decrease their representations in populations. But, the most fit schemata are more likely to be in the next generation, while the least fit are less likely. Even though Crossover and mutation can disrupt the individuals, schema patterns still exist. Holland found that a GA efficiently processes n^3 schemata in a population size of n. Eq. 7.1 describes the general growth rate of schemata in the search space.

$$m(H, t+1) = m(H, t) \cdot \textit{fitness of schemata} \tag{7.1}$$

where m is the function that describes the number of individuals that represent the schema H at generation t.

Schema theories can be viewed as macroscopic models of genetic algorithms. This means that they state something about the properties of a population in the next generation in terms of macroscopic quantities (like schema fitness, population fitness, number of individuals in a schema, etc.) measured in the current generation. Using the established methods and genetic operators of genetic algorithms, the schema with above average fitness, short defining length and lower order is more likely to survive and it increases exponentially in successive generations.

Schema theorem is an inequality that results from coarse-graining an equation for evolutionary dynamics (Eq. 7.2). It serves as the analysis tool for the GA process and states that the schema with above average fitness, short and low-order schemata is more likely to survive.

$$\langle m(H, t+1) \rangle \geq m(H, t) \frac{f(H, t)}{a_t} [1 - p] \tag{7.2}$$

[1] Schemata: plural of schema

where $\langle m(H, t+1)\rangle$ is the expected number of individuals that represent the schema H at generation $t+1$, $f(H,t)$ is the average fitness value of the individuals containing schema H at generation t and a_t is the average fitness value of the population at generation t. Eq. 7.3 defines p.

$$p = p_c \frac{\sigma(H)}{l-1} + o(H)p_m \tag{7.3}$$

where $o(H)$ is the order of the schema (the number of fixed positions in the template), ℓ is the length of the string , p_c is the crossover probability and p_m is the probability of mutation. $\sigma(H)$ is the defining length defined as the distance between the between the first and last specific positions.

Much controversy has surrounded schema theory, mainly because of its apparent lack of utility. Opponents of schema theory argue that it tells us very little about what really goes on inside an EA. Moreover, the traditional theorem of the Holland scheme is pessimistic in the sense that it only provides a lower bound on the expected growth of the schema. But many researchers and scientific have successfully proved that the Schema theory is useful and interesting study for analysis of GA. [80, 168, 169, 171, 173, 174, 176, 178, 225].

Among the direct criticisms of the Schema, Fogel and Ghozeil [99–101] criticized it for failing to estimate correctly the population of a schema in the population when fitness proportionate selection is used in problems with noise or other stochastic effects. However, this idea should be qualified for two reasons suggested by Poli [170]. First, the theorem in its original form is not applicable to problems with noise. Moreover, if the quantities involved in the schema theorems are random variables, the theorems must be interpreted as conditional statements. Second, he showed how the conditional versions of Holland and other researchers' schema theorem are indeed very useful to model the sampling of schemata in the presence of stochasticity.

Whitley's criticism [226] supposed that, because the selection becomes biased as the population grows, the observed fit of the schema changes radically. Thus, the average fitness of a schema is only relevant in the first generations of a series. Thus, the schema theorem cannot be used recursively. It is one of the most common criticisms of the Schema theorem. This criticism has been resolved in [172] and [177] in the more general setting of genetic programming. A particularly strong version of the schema theorem is the recursive conditional schema theorem, and this has solved this criticism very completely.

Schema theory has many applications. For instance, in [168], Plant and Stanton present the use of Schema theory in ergonomic research, particularly in the key areas of situation awareness and naturalistic decision making.

7.3 Previous Work on Schemata for GA

Since John Holland's work in the mid seventies on his well known schema theory [47, 116], this theory is used to explain how GAs work. More precisely, it describes how schemata are expected to propagate generation after generation under the effects of selection, crossover, and mutation.

There have been many researchers interested in studies on schema theory since Holland. Some of these where interested in giving introductions and overviews to the field of Schema Theory [80, 176, 217, 225]. For instance, in [175] , Poli reviewed the main results available to date 1997 in the theory of schemata and some recent experimental works. Then, he [176] reviewed the

main results obtained in the theory of schemata in genetic programming (GP), emphasizing their strengths and weaknesses. Another relatively recent example , White presented in his work [225] an introduction written by a mathematician and for mathematicians. In particular, he endeavored to highlight areas of the field which might be of interest to a mathematician, to point out some related open problems, and to suggest some large-scale projects.

Other authors have been interested in developing and improving this theory. Since the Holland Schema theorem gives a lower bound on the expected fraction of a population in a schema after one generation of a simple genetic algorithm, Wright gives formulas for the exact expected fraction of a population in a schema after one generation of the simple genetic algorithm [229]. In [235], the main objective was to propose a schema theory which could be a more realistic model for genetic programming and could be potentially employed for improving GP in practice.

Some versions of the Schema theorem have been proposed to held with different crossover, mutation, and selection. Riccardo Poli and others took up the challenge of generalizing the Schema theorem for genetic programming and this brought the discussion over the usefulness of the Schema theorem to the fore. In 1997, Poli [176] reviewed the main results obtained in the theory of schemata in Genetic Programming (GP) emphasising their strengths and weaknesses. Then he proposed a new, simpler definition of the concept of schema for GP which is closer to the original concept of schema in genetic algorithms (GAs). The simplicity of this definition along with that of one-point crossover has allowed him to derive a new schema theorem, since he shown that this crossover can explore the search space of programs better than standard crossover.

Cheng et al. [63] focus on a new, and general schema theory for gene expression programming, where the chromosome consists of a linear, symbolic string of fixed length composed of one or more genes. The result shows that the individuals with high fitness values, shorter schema order, more function nodes, and the shorter the insertion sequence have the greater transmission probability than the average ones.

As another research direction, many studies are in the literature that were interested in improving genetic algorithms based on schema theory. For instance, in [234], Zhou studied an improved GA based on the analysis of Schema Theory. In addition, a method was developed to study the structure of GA solution space by characterizing interactions between genes. Also, the main purpose of the research of Liu [83] is to improve the mathematical modeling of GA, and to explore how to overcome the shortcomings of traditional algorithms under the Schema theorem. The research results show that the improved GA can be more widely applied to the optimization problems and play an important role in solving practical problems. In [207], Tao and Xin analyzed the impacts of operators in artificial immune optimization algorithms on schema survival and population diversity. Therefore, a schema theorem based on artificial immune is proposed and proved. It is validated in the experiments, which provides the theoretical basis for the designs and improvements of operators in artificial immune algorithms.

The traditional schema theorem uses a binary representation. However, in reality, solving many problems using a genetic algorithm requires using other encoding methods. In [232], Yuping first gives some typical genetic operators, on the basis of which they proved the scheme theorem using limited character set encoding. The result shows that the number of schema with low order, short defined length and a fitness value greater than the average fitness value grows exponentially.

In this contribution, the schema theory is considered to study how the GAs are working using non-binary encodings, especially Lehmer code and Inverse transposition array.

7.4 Preservation of schema for Lehmer code and Inverse transposition array

In chapter 6, we studied the impact of indirect encoding on the transmission of genetic properties in permutation problems. The indirect encodings in question were the Lehmer code, the Inversion table, the Transposition array, and the Inverse transposition array. During this study, we notice specific criteria in two between these codings, the Lehmer code and the Inverse transposition array. We have observed the conservation of a part of the genetic heritage during the crossover process with the single point operator. If the parents are encoded in a Lehmer code, the offspring will retain the prefix. Conversely, if the parents are coded in the Inverse transposition array, the offspring will keep the suffix.

To explain why we have this conservation, we must observe the algorithms of each encoding and analyse it. First of all, recalling the equation of the Lehmer code from section 5.4, we have that the Lehmer code, associates a unique code $L(\pi)$ to each permutation $\pi = \pi_1\pi_2 \dots \pi_n$. $L(\pi)$ is a sequence $L(\pi) = l_1 l_2 \dots l_n$ with l_i the number of elements that are smaller than π_i and that appear to the right of π_i in the permutation.

$$l_i = Card\{j | j > i \quad \& \quad \pi_j < \pi_i\} \tag{7.4}$$

For a better observation, the process of decoding of the Lehmer code into permutation also helps to deduce the proposed conservation, see pseudocode 7.1. The main reason is that ,during decoding, the value of the i^{th} gene depends on the $(i-1)$ first positions. For example, to decode the 4^th gene, its value alone does not really tell us enough information, it depends on the value of the 1^{st}, 2^d then 3^d genes. More precisely, at each step, the value of the i^{th} gene in Lehmer code represents the position of the value of the i^{th} gene for the permutation among an alphabet list ($\{1, 2, 3, \dots n\}$ where n =length Lehmer code), then this value is removed from the list. This deletion has its consequences for the rest of the decoding.

Input: LehmerCode
Output: Permutation
l :length of Lehmer code
$Alphabet$: list of length l
for $i = 0;\ i < l;\ i++$ **do**
  $Alphabet$.add($i + 1$)
end
for $i = 0;\ i < l;\ i++$ **do**
  index=LehmerCode[i]
  Permutation[i]$\leftarrow$ $Alphabet$[index]
  $Alphabet$.remove(index)
end

Algorithm 7.1: Decoding of Lehmer code

For a larger view, assuming $P1$ and $P2$ are two parents encoded into the Lehmer code. When applying the crossover using the single point operator, assuming after the k^{th} gene, the first part of the offspring 1 ($O1$) is the same as $P1$'s first part, and the second part of $O1$ is the same as $P2$'s second part. Using the Lehmer code, the decoding of each allele depends on those that precede it. The $O1$'s alleles inherited from $P2$ will therefore be decoded according to the first part, inherited from $P1$. This will give rise to new values (not present in either $P1$ or $P2$).

For the Inverse transposition array encoding, during decoding, the value of the i^{th} gene depends on the $(n - i)$ last positions. That means that the suffix is preserved after the crossover process and the decoding into permutation.

Let take the example of two parent of length 14, $P1 = [10, 5, 9, 13, 14, 4, 8, 7, 1, 3, 6, 11, 2, 12]$ and $P2 = [7, 4, 6, 5, 3, 1, 9, 11, 10, 14, 8, 12, 13, 2]$. In Fig. 7.1 and 7.2, we see how the prefix, respectively the suffix, is preserved for the LC, respectively ITA, depending on the position of the cutting point. The colser the position of this point is to 14, the more schema is preserved.

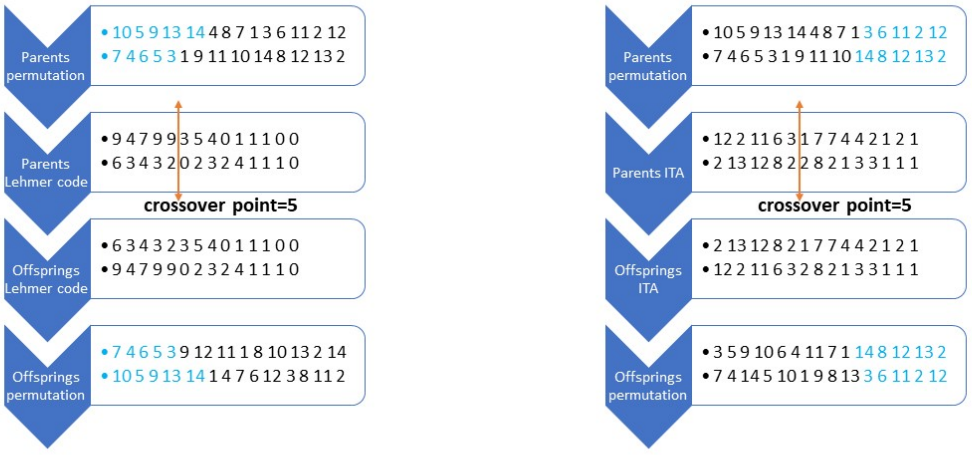


Figure 7.1: Schema preservation using LC and ITA with a crossover point after the 5^{th} gene

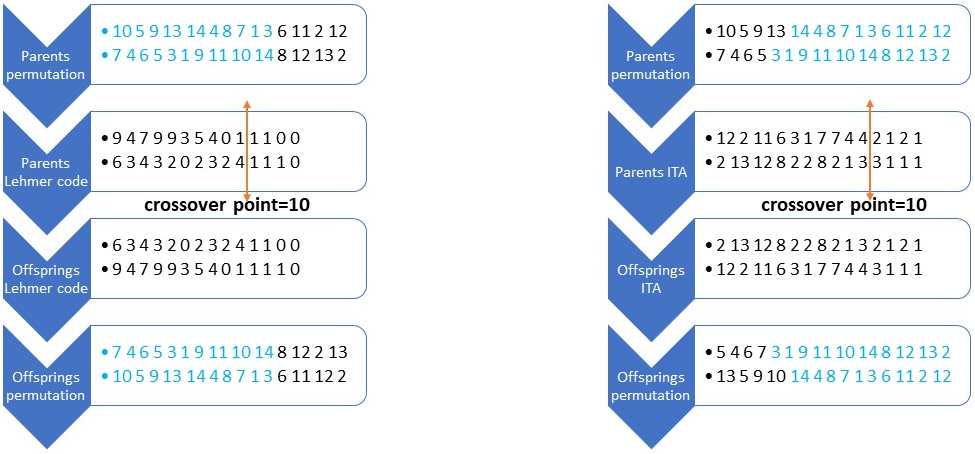


Figure 7.2: Schema preservation using LC and ITA with a crossover point after the 10^{th} gene

7.5 Proposed method: schema preservation based on crossover point selection

During an optimization process, it is important to keep the interesting parts of good solutions. This is what the recombination (which leads to the generation of new solutions) is supposed to achieve. More precisely, it is the crossover which should play a part in this transmission process. Indeed, the crossover is an *exploitation* operator. Exploitation means the intensification of the search within good solutions' neighborhood. It is therefore necessary to ensure that the genetic

properties of the best individuals are preserved. However, as we noted in Sections 6.3.2 and 6.3.3, during the crossover step using indirect encoding, some parents' genetic properties are lost.

Among the four indirect encoding studied in Chapter 6, we can note that two among them have particular attributes. Applying the 1-point crossover to the Lehmer code preserves the prefix (alleles located before the crossover point) and the Inverse transposition array preserves the suffix (alleles located after the crossover point). This is illustrated by Fig. 7.1 and 7.2. This preservation is also noticeable after the decoding phase. That is to say that we observe this conservation in the genotypes as well. Indeed, the decoding process of the Lehmer code is based on the values of the first genes, while the Inverse transposition array's decoding is based on the last genes.

The idea behind our contribution is as follows: the preservation of a large number of alleles for good solutions should be stimulated. In the case of the Lehmer code, one should seek the preservation of the begin of the TSP's tour, while for the Inverse transposition array, the end of the tour will be kept.

To favour this preservation of the parents' genetic properties, we consider to select the crossover point (denoted Xp) according to the parent solutions' quality. Let f be the objective function value of the best of the two selected parents and f_{min} (respectively f_{max}) be the smallest (respectively the largest) objective function value within the current population (parents' population). We define σ as the devaluation rate of f_{min} by f (see Eq. 7.5).

$$\sigma = \frac{f - f_{min}}{f_{max} - f_{min}} \tag{7.5}$$

The smaller σ, the better the quality of parents (at least one of them) within the population. Therefore, the crossover point must be chosen as a function of σ (see Eq. 7.6). For parents of good quality, we will try to choose a cutoff point as far as possible. Conversely, when the parents are of poor quality, the crossover by will be selected after the very first genes, in order to create a strong disruption that can lead to the generation of good offspring.

$$Xp = exp(-\sigma) \times n \tag{7.6}$$

where n is the number of decision variables.

7.6 Experiments

Our objective is, first of all, to study the transmission of characteristics from parents to offspring when using the adaptive crossover, and to compare it with the classical version. For this, we use the metrics that focus on the similarities between the solutions (Section 7.6.1). Then, we evaluate and compare the quality of the results of each encoding with respect to the objective function with and without using the adaptive crossover, (Section 7.6.2). For this study, we considered the `eil51` problem (a traveling salesman problem instance of 51 cities proposed in the TSPLIB [213]). The experiment parameters used are summarized in Table 7.1.

The results presented in this section correspond to the averages of 10 independent runs.

Encoding	Crossover	Mutation	Population size	Nb. of generations
Lehmer Code	Single point	BitFlip	200	300
Inverse transposition array	Single point	BitFlip	200	300

Table 7.1: Experiment parameters

7.6.1 Assessment of transmissions from parents to offspring

The goal is to evaluate the transmission of the genetic characteristics from parents to offspring in order to observe the impact of the adaptation of the choice of the crossover point for each encoding. For this purpose, several indicators are used: the Hamming distance, the edge based indicator and the position based indicator.

The first indicator is the Hamming distance that measures the similarity between two solutions with respect to the decision variables.

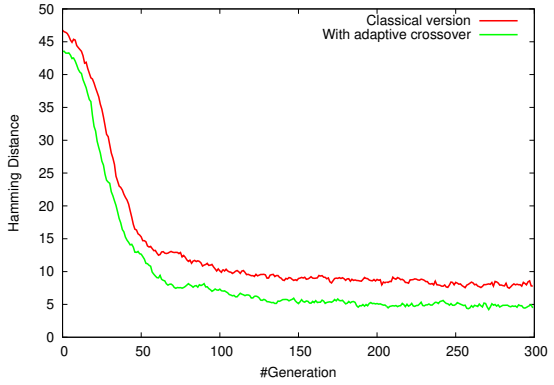


Figure 7.3: Hamming Distance using Lehmer code

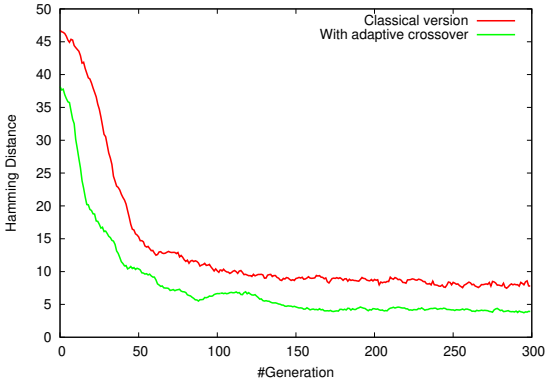


Figure 7.4: Hamming Distance using Inverse transposition array

Fig. 7.3 shows the variation of the value of Hamming distance in function of generation for the LC with and without adaptation. We can see that we have lower values of HD with adaptive crossover. Fig. 7.4 shows this variation for ITA where we also have the same interpretation. The use of adaptation during crossover made it possible to conserve the alleles of the parents. The conservation is strong for the best individuals. Because these best individuals are the ones who survive and reproduce, this preservation will therefore be present over the generations.

The second indicator is the Edge Based Indicator (EBI) [81] that counts the edges which are present both in parent and in offspring. It is defined by Eq. 5.8 in chapter 5. Fig. 7.5 shows the variation of the value of EBI in function of generation for the LC with and without adaptation. Fig. 7.6 shows this variation for ITA.

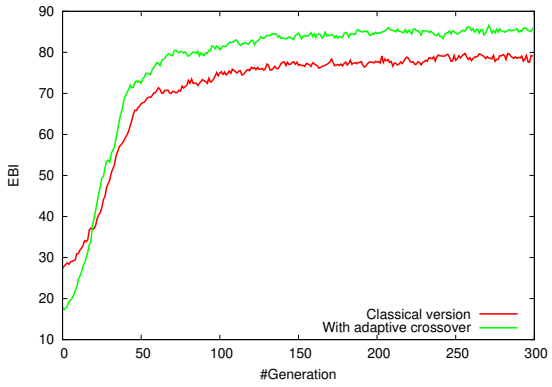


Figure 7.5: EBI using Lehmer code

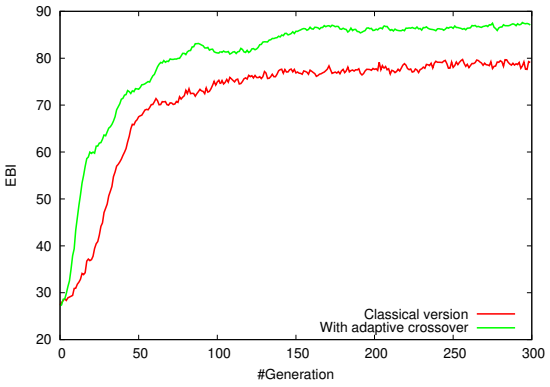


Figure 7.6: EBI using Inverse transposition array

The last indicator is the Position Based Indicator (PBI) [81], it is a metric that looks at the position of cities among parents and children.

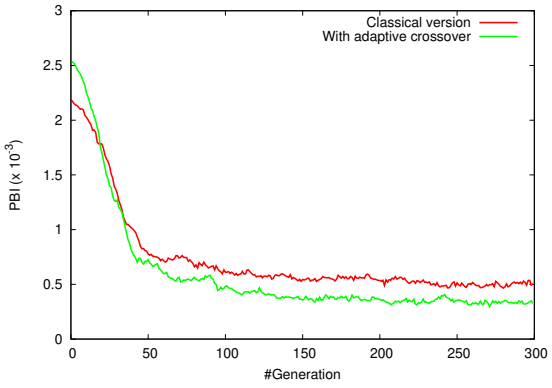


Figure 7.7: PBI using Lehmer code

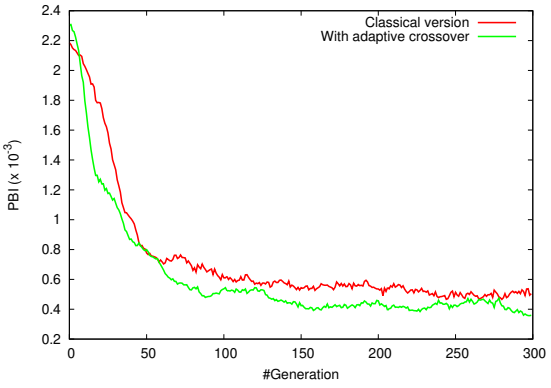


Figure 7.8: PBI using Inverse transposition array

Fig. 7.7 shows the variation of the value of PBI in function of generation for the Lehmer code with and without adaptation on the crossover operator. Fig. 7.8 shows this variation for Inverse transposition array.

The two indicators, PBI and EBI, are calculated from the decoded solutions. The decoding process for the Lehmer code (respectively for the Inverse transposition array) is based on alleles located on the left (respectively on the right), i.e. the prefix (respectively the suffix). Thus, given that the adaptation operator preserves the prefix / suffix, it favors the preservation of edges and positions.

7.6.2 Analysis of fitness distribution

In this section, we study the evolution of the fitness value over generations and compare the performance of the two different types of encoding with and without the adaptation. This is done using the method presented in section 6.4.1.

The principle of choosing the crossover point according to the values of the objective functions aims to reinforce the "exploitation" pressure. However, an uncontrolled exploitation could lead to a loss of diversity and to premature convergence [3, 205]. Fig. 7.9 and 7.10 show a distribution of solutions within the different groups which is almost similar to the version without adaptation (see

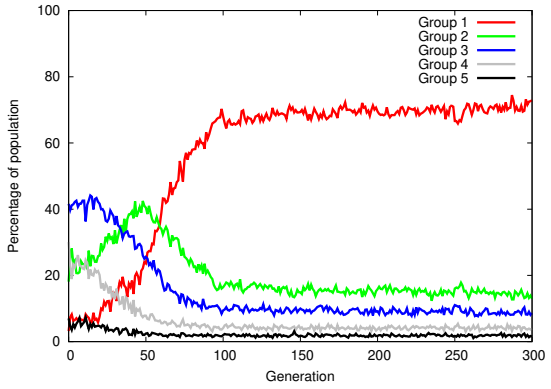


Figure 7.9: Classification using Lehmer code with adaptation

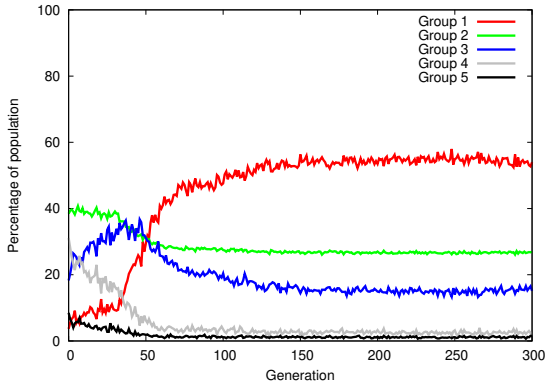


Figure 7.10: Classification using Inverse transposition array encoding with adaptation

Fig. 6.15 and 6.18). This could be explained by the fact that the pressure to conserve the genetic characteristics of the best solutions being exerted at the crossover level and not during the selection step, the solutions of poor quality can participate to the recombination process. Therefore, the proposed adaptation would act in this case as an exploration tool by creating a disruption when generating offspring (weak inheritance from the parents). This could preserve diversity and avoid premature convergence.

7.7 Conclusion

In 1975, Holland developed a schema theory to understand how genetic algorithms (GAs) work in a formal sense. Specifically, this theory describes how schema are suppose to propagate from generation to generation under the effects of selection, crossover, and mutation. This chapter has studied the transmission of genetic properties in an optimization process using two indirect encodings, the Lehmer code (LC) and the Inverse transposition array (ITA). The choice of these encoding is made because the LC decoding process (respectively for ITA) was characterized by preservation of the prefix (respectively the suffix). Thus, an adaptation operator has been proposed which preserves the prefix / suffix of good solutions. This adaptation has favored the preservation of genetic characteristics such as edges and positions.

Chapter 8

Conclusion

During this thesis, we have made contributions in two fields theoretical computer science, which are on the one hand enumerative combinatorics and on the other hand evolutionary computation. Our goal was to study the combinatorial properties of some combinatorial objects, and, as a complementary work, to study the application of combinatorial objects properties (permutations) in evolutionary computation.

To conclude this work, we propose in section 8.1 a synthesis allowing to provide some answers to the problems of this study. Some perspectives relating to our contributions are then given in the section 8.2

8.1 Summary

First, we looked at a recent line of research on classical pattern avoidance on words subject to certain growth restrictions. So after Baril et al. [23] studied the distribution of descents on sets of Catalan words avoiding a pattern of length at most 3, we studied, in Chapter 3, the connections between sequences on the Online encyclopedia of Integer sequences [213] and Catalan words avoiding two patterns of length 3. More precisely:

- In this study, there was a lot of trivial cases, such as *Superfluous patterns*, *Ultimately constant sequences* and the *Counting sequence* n,
- Many cases were enumerated by counting via recurrence, mainly via the counting sequence $2(n-1)$, sequences involving 2^n, sequences involving binomial coefficients and sequences involving Fibonacci(-like) numbers,
- It is important to note that for some cases, the sequences have not been studied in the literature, so they were counted giving bivariate generating functions $C_\pi(x, y)$ where the coefficient of $x^n y^k$ is the number of Catalan words of length n having k descents and avoiding π.

Also some propositions and corollaries were proposed to match some cases, by using ascent sequences. We specially used for the enumerations the corollary 1 that says, for $n \geq 4$ and a pattern π of length three, $\mathcal{A}_n(\pi) = \mathcal{C}_n(\pi)$ if and only if $\pi \in \{001, 010, 012, 102\}$.

Then, the study of sorting permutation (arranging permutations in increasing order) was in the top of our research interests. More precisely, we studied a variant of pattern-avoiding machines where the first stack avoids (σ, τ), a pair of patterns of length three. Following [60], we call it (σ, τ)-machine. More specifically, we restrict ourselves to those pairs of patterns for which sortable permutations are counted by either the Catalan numbers or two of their close relatives: the binomial transform of Catalan numbers and the Schröder numbers.

- For the pair $(132, 231)$ we showed that sortable permutations are those avoiding 1324 and 2314, a set whose enumeration is given by the large Schröder numbers.
- Under certain conditions on the avoided patterns, the output of the first stack is bijectively related to its input (see [33,59]): it follows that for three pairs of patterns, namely $(123, 213)$, $(132, 312)$ and $(231, 321)$, sortable permutations are counted by the Catalan numbers. This result was proved independently in [21, 33].
- For the pair $(123, 132)$, we proved that sortable permutations are those avoiding the patterns 2314, 3214, 4213 and the generalized pattern $[24\bar{1}3$. Also, we proved that sortable permutations are enumerated by the Catalan numbers by showing that the distribution of the first element is given by the well-known Catalan triangle.
- We showed that for the pair $(123, 312)$ the corresponding counting sequence is the binomial transform of Catalan numbers.

In the second part of the thesis, we investigated one of the most interesting and difficult permutation problems, the Transport Salesman Problem (TSP). This difficulty is due in part to the combinatorial explosion that occurs when the size of the problems increases. To solve such problems, we were interested in metaheuristic optimization methods, in particular the genetic algorithm, inspired by Darwin's theory. First, our objective was to observe the transmission of genetic properties during the optimization algorithm for the TSP problem. Different encoding types have been studied to represent permutations, in order to avoid generating duplicates during the crossover and mutation steps. Among these encodings, some were introduced previously like the Lehmer code and the Inversion table and others are new in the literature like the Transposition array and the Inverse transposition array. The work carried out during in this section is summarized in the following points:

- In order to evaluate the transmission of the genetic characteristics from parents to offspring to observe the impact of the encoding, several indicators were used: the Hamming distance, the edge based indicator and the position based indicator. The results of experiments was explained by the diversity of individuals in the cases of indirect encodings,
- To study the evolution of the fitness value over generations and to compare the performance of the different types of encoding, a method was used and it was inspired by the set of experimentation introduced by Portman and Vignier in [183]. They classify solutions, for each generation, in five groups based on the fitness values. The results of experiments has shown that for the indirect encoding, greater diversity is observed because the solutions of group 1 correspond to the majority, but the proportions of the solutions in the other groups are not negligible.

Then, the objective of the next contribution was is to study the transmission of genetic properties when using indirect encodings that preserve schemata. After finding that there is conservation

properties of the suffix or prefix for the Lehmer code and the Inverse transposition array, this led us to the studies on the schema theory of Holland. He realized that when the genetic algorithm evaluates the fitness of a solution in the population, it is in fact evaluating the fitness of many solutions in an implicitly parallel mode. The work carried out during in this section is summarized in the following points:

- The preservation of a large number of alleles for good solutions is stimulated by applying an adaptation on the choice of the cutting point during the crossover process according to the parent solutions' quality.
- The use of adaptation during crossover made it possible to retain more alleles from the best individuals. Because these best individuals are those who survive and reproduce, this preservation will therefore be present over generations. This has been shown using Hamming Distance, Edge Based Indicator, and Position Based Indicator.
- We have studied the evolution of the fitness value over generations after applying the adaptation method on the crossover operator. It shows a distribution of the solutions within the different groups almost similar to the version without adaptation. This means the way exploitation is forced in our proposal did not to diversity loss.

8.2 Future research

To conclude we offer some open questions and perspectives offered by this work.

The first future research suggestion concerns Catalan words. They are in bijection with Dyck paths and thus pattern avoiding Catalan words correspond to restricted Dyck paths. For instance, a Catalan word avoiding 012 corresponds to a Dyck path of height at most two. In this context, it can be of interest to investigate how our results on pattern avoiding Catalan words translate to corresponding restricted Dyck paths.

Even if in this thesis we restrict ourselves to the avoidance of two patterns of length 3, some classes considered here can be trivially extended to larger length patterns, for instance $\mathcal{C}(102, 201) = \mathcal{C}(01012, 01201)$. In this light, it can be of interest to explore Catalan words avoiding patterns of length 4 or more, triples of patterns or generalized patterns.

Our contribution in stack-sorting permutation can open to many other studies and researches about stack-sorting. More precisely, we can apply other restrictions on the stack, other than pattern avoidance, or we can choose to apply constraints on the Stack-sorting algorithm itself.

The restrictions on the stack are expressed by saying that, at each step of the execution, the elements into the stack (read from top to bottom) must avoid certain forbidden configurations. In particular, k-interval avoidance, k-length avoidance, k-ecart avoidance ... For instance, to avoid k-length, at every step, the sequence of numbers contained in the stack has to contain intervals of length less than k.

As a constraint on the Stack-sorting algorithm, we suppose the algorithm we perform on the stack is equivalent to make a short-circuit through a stack. This can be applied to perform a right greedy algorithm and avoid the stack and the pop steps of an element that can be sorted directly. The fact that we choose to use the short circuit step does not change anything in the algorithm concerning stacking and arranging, but when we want to study some properties and restrictions, it can affect.

For example, if we take the example of the permutation 652143, the number of intervals and their length differs if we allowed the short circuit or if we do not.

There are many ways to broaden the study of the inheritance of genetic properties in permutation problems. In our contributions we have focused on the popular and challenging optimization problem, the traveling salesman problem (TSP). The same experiments can be done using other permutation problems, such as Asymmetric traveling salesman problem (ATSP), Sequential ordering problem (SOP), Capacitated vehicle routing problem (CVRP),... These problems are challenging since they introduce new constraints.

There is another idea that can be interesting in exploiting good solutions. If one has a priori knowledge of the problem and perhaps of the optimal solution, this can lead to a preliminary sorting on the alphabet of the problem in question. The issue here is also whether we can also have an impact on the quality of the solutions after having forced this procedure.

While exploiting the schema preservation in the genetic algorithm, the focus was on the single point crossover operator, but as further work we can seek preservation of the schema when using k-points crossover. This can be considered as another challenging and interesting research track since it would be necessary to find other mechanisms ensuring the preservation.

Bibliography

[1] jMetal web site. http://jmetal.sourceforge.net/.

[2] The on-line encyclopedia of integer sequences. http://oeis.org.

[3] W. Abdou, C. Bloch, D. Charlet, and F. Spies. Adaptive multi-objective genetic algorithm using multi-pareto-ranking. In *GECCO 2012, 14th Int. Genetic and evolutionary computation Conference*, pages 449 – 456, Philadelphia, Pennsylvania, United States, jul 2012.

[4] M. Aigner. Motzkin numbers. *European Journal of Combinatorics*, 19(6):663–675, 1998.

[5] S. Akter, M. Murad, R. Chaity, M. Sadiquzzaman, and S. Akter. Genetic algorithm with updated multipoint crossover technique and its application to tsp. In *2020 IEEE Region 10 Symposium (TENSYMP)*, pages 1209–1212, 2020.

[6] S. Akter, N. Nahar, M. ShahadatHossain, and K. Andersson. A new crossover technique to improve genetic algorithm and its application to tsp. In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pages 1–6, 2019.

[7] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, 2002.

[8] M. Albert and M. Atkinson. Sorting with a forklift. *Electron. J. Comb.*, 9, 2002.

[9] M. Albert, M. Atkinson, and S. Linton. Permutations generated by stacks and deques. *Annals of Combinatorics*, 14(1):3–16, 2010.

[10] M. Albert, M. Bouvel, and V. Féray. Two first-order logics of permutations. *Journal of Combinatorial Theory, Series A*, 171:105158, 2020.

[11] A. Asinowski, A. Bacher, C. Banderier, and B. Gittenberger. Analytic combinatorics of lattice paths with forbidden patterns, the vectorial kernel method, and generating functions for pushdown automata. *Algorithmica*, 82(3):386–428, 2019.

[12] M. Atkinson, M. Murphy, and N. Ruškuc. Sorting with two ordered stacks in series. *Theoretical Computer Science*, 289(1):205–223, 2002.

[13] E. Babson and E. Steingrímsson. Generalized permutation patterns and a classification of the mahonian statistics. *Séminaire Lotharingien de Combinatoire [electronic only]*, 44:B44b, 18 p.–B44b, 18 p., 2000.

[14] C. Banderier and P. Flajolet. Basic analytic combinatorics of directed lattice paths. *Theoretical Computer Science*, 281(1-2):37–80, 2002.

[15] J. Bandlow, E. Egge, and K. Killpatrick. A weight-preserving bijection between Schröder paths and Schröder permutations. *Annals of Combinatorics*, 6(3):235–248, 2002.

[16] E. Barcucci, A. Del Lungo, E. Pergola, and R. Pinzani. Eco:a methodology for the enumeration of combinatorial objects. *Journal of Difference Equations and Applications*, 5(4-5):435–490, 1999.

[17] E. Barcucci, A. D. Lungo, E. Pergola, and R. Pinzani. Some combinatorial interpretations ofq-analogs of Schröder numbers. *Annals of Combinatorics*, 3(2-4):171–190, 1999.

[18] E. Barcucci, R. Sprugnoli, and R. Pinzani. The Motzkin family. *Pure Mathematics and Applications*, 2(3-4):249–279, 1992.

[19] J.-L. Baril. Gray code for permutations with a fixed number of cycles. *Discrete Mathematics*, 307:1559–1571, 06 2007.

[20] J.-L. Baril. Statistics-preserving bijections between classical and cyclic permutations. *Information Processing Letters*, 113(1):17–22, 2013.

[21] J.-L. Baril, G. Cerbai, C. Khalil, and V. Vajnovszki. Catalan and Schröder permutations sortable by two restricted stacks, 2020.

[22] J.-L. Baril, C. Khalil, and V. Vajnovszki. Catalan words avoiding pairs of length three patterns, 2021.

[23] J.-L. Baril, S. Kirgizov, and V. Vajnovszki. Descent distribution on Catalan words avoiding a pattern of length at most three. *Discrete Mathematics*, 341(9):2608–2615, 2018.

[24] J.-L. Baril, S. Kirgizov, and V. Vajnovszki. Descent distribution on Catalan words avoiding a pattern of length at most three. *American Discrete Mathematics*, pages 2608–2615, 2018.

[25] J.-L. Baril and A. Petrossian. Equivalence classes of Dyck paths modulo some statistics. *Discret. Math.*, 338:655–660, 2015.

[26] J.-L. Baril and A. Petrossian. Equivalence classes of Dyck paths modulo some statistics. *Discrete Mathematics*, 338(4):655–660, 2015.

[27] J.-L. Baril and A. Petrossian. Equivalence classes of Motzkin paths modulo a pattern of length at most two. *Journal of Integer Sequences*, 2015.

[28] R. Barr, R. Helgason, and J. Kennington. *Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies*. Springer Publishing Company, Incorporated, 2012.

[29] R. Basu, A. Bose, S. Ganguly, and R. Hazra. Spectral properties of random triangular matrices, 2012.

[30] A. Baxter and L. Pudwell. Ascent sequences avoiding pairs of patterns. *The Electronic Journal of Combinatorics*, 22(1), 2015.

[31] S. Bekiroğlu, T. Dede, and Y. Ayvaz. Implementation of different encoding types on structural optimization based on adaptive genetic algorithm. *Finite Elements in Analysis and Design*, pages 826–835, 2009.

[32] R. Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9(1):61–63, Jan. 1962.

[33] K. Berlow. Restricted stacks as functions, 2020.

[34] J. Berstel and D. Perrin. *Theory of codes*. Academic Press, 1985.

[35] N. Biggs, E. K. Lloyd, and R. J. Wilson. *Graph Theory, 1736-1936*. Clarendon Press, USA, 1986.

[36] S. Billey, W. Jockusch, and R. P. Stanley. Some combinatorial properties of schubert polynomials. *Journal of Algebraic Combinatorics 2*, pages 345–374, 1993.

[37] M. Bòna. A simplicial complex of 2-stack sortable permutations. *Advances in Applied Mathematics*, 29(4):499–508, 2002.

[38] M. Bòna. A survey of stack-sorting disciplines. *The Electronic Journal of Combinatorics*, 9(2), 2003.

[39] M. Bòna. *Combinatorics of Permutations*. CRC Press, Inc., USA, 2004.

[40] M. Bòna. *A walk through combinatorics: an introduction to enumeration and graph theory*. World Scientific Pub., 2006.

[41] M. Bòna and R. Flynn. The average number of block interchanges needed to sort a permutation and a recent result of Stanley. *Information Processing Letters*, 109(16):927–931, 2009.

[42] M. Bousquet-Mélou. Multi-statistic enumeration of two-stack sortable permutations. *Electron. J. Comb.*, 5, 1998.

[43] M. Bousquet-Mélou, A. Claesson, M. Dukes, and S. Kitaev. (2 + 2)-free posets, ascent sequences and pattern avoiding permutations. *Journal of Combinatorial Theory, Series A*, 117(7):884–909, 2010.

[44] M. Bousquet-Mélou. Sorted and/or sortable permutations. *Discrete Mathematics*, 225(1):25–50, 2000. FPSAC'98.

[45] M. Bouvel and O. Guibert. Refined enumeration of permutations sorted with two stacks and a d_8-symmetry, 2012.

[46] C. Brennan and S. Mavhungu. Peaks and valleys in Motzkin paths. *Quaestiones Mathematicae*, 33(2):171–188, 2010.

[47] C. Bridges and D. Goldberg. An analysis of reproduction and crossover in a binary-coded genetic algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, page 9–13, USA, 1987. L. Erlbaum Associates Inc.

[48] P. Brucker. *Scheduling Algorithms*. Springer Publishing Company, Incorporated, 5th edition, 2010.

[49] P. Brändén and A. Claesson. Mesh patterns and the expansion of permutation statistics as sums of permutation patterns. *The Electronic Journal of Combinatorics*, 18(2), 2011.

[50] R. Burkard and E. Cela. Linear assignment problems and extensions, 1998.

[51] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems. Revised reprint.* SIAM - Society of Industrial and Applied Mathematics, 2012. 393 Seiten.

[52] A. Burstein. *Enumeration of words with forbidden patterns*. University of Pennsylvania, 1998.

[53] A. Burstein and I. Lankham. Combinatorics of patience sorting piles. *FPSAC Proceedings 2005 - 17th Annual International Conference on Formal Power Series and Algebraic Combinatorics*, 07 2005.

[54] A. Burstein and T. Mansour. Words restricted by 3-letter generalized multipermutation patterns. *Annals of Combinatorics*, 7:1–14, 2001.

[55] A. Burstein and T. Mansour. Words restricted by patterns with at most 2 distinct letters. *The Electronic Journal of Combinatorics*, 9(2), 2002.

[56] Y. Cai and C. Yan. Counting with Borel's triangle, 2018.

[57] L. Campbell, S. Dahlberg, R. Dorward, J. Gerhard, T. Grubb, C. Purcell, and B. Sagan. Restricted growth function patterns and statistics. *Discrete Mathematics and Theoretical Computer Science*, 18(2):electronic, 2016.

[58] E. Cantú-Paz and D. Goldberg. Efficient parallel genetic algorithms: theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 186(2):221–238, 2000.

[59] G. Cerbai. Sorting cayley permutations with pattern-avoiding machines, 2020.

[60] G. Cerbai, A. Claesson, and L. Ferrari. Stack sorting with restricted stacks, 2019.

[61] G. Cerbai, A. Claesson, L. Ferrari, and E. Steingrímsson. Sorting with pattern-avoiding stacks: the 132-machine, 2020.

[62] V. Chahar, S. Katoch, and S. Chauhan. A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*, 80, 02 2021.

[63] H. Cheng, J. Zhao, and H. Wang. Schema theorem based on probability for multigenic chromosomes genes expression programming. *International journal of security and its applications*, 10(1):87–94, 2016.

[64] J. Chung, S. Oh, and I. Choi. A hybrid genetic algorithm for train sequencing in the korean railway. *Omega*, 37(3):555–565, 2009.

[65] A. Claesson, M. Dukes, and E. Steingrímsson. Permutations sortable by $n - 4$ passes through a stack, 2009.

[66] A. Claesson and S. Linusson. $n!$ matchings, $n!$ posets. *Proceedings of the American Mathematical Society*, 139(02):435–435, 2011.

[67] P. Consoli and L. Minku. Dynamic selection of evolutionary algorithm operators based on online learning and fitness landscape metrics. volume 20, pages 359–370, 12 2014.

[68] J. Contreras, P. Bosch, M. Varas, and F. Basso. A new genetic algorithm encoding for coalition structure generation problems. *Mathematical Problems in Engineering*, page 1–13, 2020.

[69] R. Cori, B. Jacquard, and G. Schaeffer. Description trees for some families of planar maps. In *Proceedings of the 9th Conference on Formal Power Series and Algebraic Combinatorics*, pages 196–208, 1997.

[70] S. Corteel, M. Martinez, C. Savage, and M. Weselcouch. Patterns in inversion sequences I, 2016.

[71] G. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958.

[72] E. Czabarka, R. Flórez, L. Junes, and J. Ramírez. Enumerations of peaks and valleys on non-decreasing Dyck paths. *Discrete Mathematics*, 341(10):2789–2807, 2018.

[73] K. Dahal and J. McDonald. Generational and steady-state genetic algorithms for generator maintenance scheduling problems. In *Artificial Neural Nets and Genetic Algorithms*, pages 259–263, Vienna, 1998. Springer Vienna.

[74] G. Dantzig, D. R. Fulkerson, and M. Selmer. *Solution of a Large-Scale Traveling-Salesman Problem*. RAND Corporation, Santa Monica, CA, 1954.

[75] K. Deep and M. Thakur. A new mutation operator for real coded genetic algrithms. *Applied Mathematics and Computation*, 193:211–230, 10 2007.

[76] C. Defant. Counting 3-stack-sortable permutations. *Journal of Combinatorial Theory, Series A*, 172:105209, 2020.

[77] C. Defant and K. Zheng. Stack-sorting with consecutive-pattern-avoiding stacks, 2020.

[78] D. Desantis, R. Field, W. Hough, B. Jones, R. Meissen, and J. Ziefle. Permutation Pattern Avoidance and the Catalan Triangle. *Missouri Journal of Mathematical Sciences*, 25(1):50 – 60, 2013.

[79] E. Deutsch. Dyck path enumeration. *Discrete Mathematics*, 204(1-3):167–202, 1999.

[80] M. Dianati, I. Song, and M. Treiber. An introduction to genetic algorithms and evolution. 2002.

[81] L. Djerid, M.-C. Portmann, and P. Villon. Performance analysis of permutation crossover genetic operators. *Journal of Decision Systems*, page 157–177, 1996.

[82] R. Donaghey and L. Shapiro. Motzkin numbers. *Journal of Combinatorial Theory, Series A*, 23(3):291–301, 1977.

[83] L. Donghai. Mathematical modeling analysis of genetic algorithms under schema theorem. *Journal of Computational Methods in Sciences and Engineering*, 19:1–7, 05 2019.

[84] D. Drake and R. Gantner. Generating functions for plateaus in Motzkin paths. *Journal of the Chungcheong Mathematical Society*, 25(3):475–489, 2012.

[85] Z. Drezner. Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem. *Computers & Operations Research*, 35(3):717–736, 2008. Part Special Issue: New Trends in Locational Analysis.

[86] P. Duchon. On the enumeration and generation of generalized Dyck words. *Discrete Mathematics*, 225(1-3):121–135, 2000.

[87] M. Dukes, S. Kitaev, J. Remmel, and E. Steingrímsson. Enumerating $(\mathbf{22})$-free posets by indistinguishable elements. *Journal of Combinatorics*, 2(1):139–163, 2011.

[88] M. Dukes and R. Parviainen. Ascent sequences and upper triangular matrices containing non-negative integers. *The Electronic Journal of Combinatorics*, 17(1), 2010.

[89] S. Dulucq, S. Gire, and O. Guibert. A combinatorial proof of J. West's conjecture. *Discrete Mathematics*, 187(1):71–96, 1998.

[90] S. Dulucq, S. Gire, and J. West. Permutations with forbidden subsequences and nonseparable planar maps. *Discrete Mathematics*, 153(1):85–103, 1996. Proceedings of the 5th Conference on Formal Power Series and Algebraic Combinatorics.

[91] P. Duncan and E. Steingrímsson. Pattern avoidance in ascent sequences. *The Electronic Journal of Combinatorics*, 18(1):P226, electronic, 2011.

[92] J. Durillo and A. Nebro. jMetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, 2011.

[93] R. Ehrenborg and E. Steingrímsson. The excedance set of a permutation. *Advances in Applied Mathematics*, 24(3):284–299, 2000.

[94] L. Euler and G. Kowalewski. *Institutiones calculi differentialis*. B.G. Teubneri, 1913.

[95] S. Even and A. Itai. Queues, stacks and graphs. *Theory of Machines and Computations*, page 71–86, 1971.

[96] M. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

[97] D. Foata. On the Netto inversion number of a sequence. *Proceedings of the American Mathematical Society*, 19(1):236–236, 1968.

[98] D. Foata and M. Schützenberger. Major index and inversion number of permutations. *Mathematische Nachrichten*, 83(1):143–159, 1978.

[99] D. Fogel and A. Ghozeil. Schema processing under proportional selection in the presence of random effects. *IEEE Trans. Evol. Comput.*, 1:290–293, 1997.

[100] D. Fogel and A. Ghozeil. The schema theorem and the misallocation of trials in the presence of stochastic effects. In *Proceedings of the 7th International Conference on Evolutionary Programming VII*, EP '98, page 313–321, Berlin, Heidelberg, 1998. Springer-Verlag.

[101] D. Fogel and A. Ghozeil. Schema processing, proportional selection, and the misallocation of trials in genetic algorithms. *Inf. Sci.*, 122:93–119, 2000.

[102] M. Gen and R. Cheng. Genetic algorithms and engineering design. 1997.

[103] M. Gen, R. Cheng, and L. Lin. *Network Models and Optimization: Multiobjective Genetic Algorithm Approach*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[104] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[105] D. Goldberg and R. Lingle. Alleleslociand the traveling salesman problem. In *Proceedings of the 1st International Conference on Genetic Algorithms*, page 154–159, USA, 1985. L. Erlbaum Associates Inc.

[106] R. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275 – 278, 1958.

[107] M. Goossens, F. Mittelbach, and A. Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[108] I. Goulden and J. West. Raney paths and a combinatorial relationship between rooted nonseparable planar maps and two-stack-sortable permutations. *J. Comb. Theory Ser. A*, 75(2):220–242, Aug. 1996.

[109] J. Grefenstette. Deception considered harmful. In *FOGA*, 1992.

[110] K. Grygierek. Optimization of trusses with self-adaptive approach in genetic algorithms. *Architecture Civil Engineering Environment*, 2016.

[111] M. Haj-Rachid, W. Ramdane-Cherif, P. Chatonnay, and C. Bloch. Comparing the performance of genetic operators for the vehicle routing problem. *IFAC Proceedings Volumes*, 43(17):313–319, 2010.

[112] P. Hansen and L. Kaufman. A primal-dual algorithm for the three-dimensional assignment problem. *Cahiers du CERO*, 15:327–336, 1973.

[113] M. Held and R. Karp. A dynamic programming approach to sequencing problems. In *Proceedings of the 1961 16th ACM National Meeting*, ACM '61, page 71.201–71.204, New York, NY, USA, 1961. Association for Computing Machinery.

[114] S. Heubach and T. Mansour. *Combinatorics of Compositions and Words: Solutions Manual*. Chapman & Hall/CRC, 2009.

[115] J. Holland. Outline for a logical theory of adaptive systems. *J. ACM*, 9(3):297–314, 1962.

[116] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975. second edition, 1992.

[117] J. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[118] S. Johnson. *Optimal Two- and Three-Stage Production Schedules with Setup Time Included*. RAND Corporation, Santa Monica, CA, 1953.

[119] H. Kargupta, K. Deb, and D. Goldberg. Ordering genetic algorithms and deception. In *PPSN*, 1992.

[120] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. In M. A. Fischler and O. Firschein, editors, *Readings in Computer Vision*, pages 606–615. Morgan Kaufmann, San Francisco (CA), 1987.

[121] S. Kitaev. Miscellaneous on patterns in permutations and words. *Monographs in Theoretical Computer Science. An EATCS Series Patterns in Permutations and Words*, page 317–360, 2011.

[122] S. Kitaev. *Patterns in Permutations and Words*. Springer, 2011.

[123] S. Kitaev and J. Remmel. Quadrant marked mesh patterns in alternating permutations ii. *Journal of Combinatorics*, 4(1):31–65, 2013.

[124] D. Knuth. *The Art of Computer Programming, volume 1, Fundamental Algorithms*. Addison-Wesley, 1968.

[125] D. Knuth. *The art of computer programming*. Addison-Wesley, 1973.

[126] D. Knuth. Computer programming as an art. *ACM Turing Award Lectures*, 1974.

[127] Z. Konfrst. Parallel genetic algorithms: advances, computing trends, applications and perspectives. In *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, pages 162–, 2004.

[128] B. Koohestani. A crossover operator for improving the efficiency of permutation-based genetic algorithms. *Expert Systems with Applications*, 151:113381, 2020.

[129] T. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. Cowles Foundation Discussion Papers 4, Cowles Foundation for Research in Economics, Yale University, 1955.

[130] C. Krattenthaler. Permutations with restricted patterns and Dyck paths. *Advances in Applied Mathematics*, 27(2):510–530, 2001.

[131] C. Krattenthaler and S. G. Mohanty. Lattice path combinatorics – applications to probability and statistics.

[132] D. Kreher and D. Stinson. *Combinatorial Algorithms: Generation, Enumeration, and Search*. Discrete Mathematics and Its Applications. Taylor & Francis, 1998.

[133] D. Kremer and W. Shiu. Finite transition matrices for permutations avoiding pairs of length four patterns. *Discrete Mathematics*, 268(1-3):171–183, 2003.

[134] R. Kumar. Using new variation crossover operator of genetic algorithm for solving the traveling salesmen problem in e-governance.

[135] A. Land and A. Doig. *An Automatic Method for Solving Discrete Programming Problems*, volume 28, pages 105–132. 11 2010.

[136] P. Larranaga, C. Kuijpers, R. Murga, I. Inza, and S. Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13:129–170, 01 1999.

[137] D. Lehmer. Teaching combinatorial tricks to a computer. 1960.

[138] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.*, 21(2):498–516, Apr. 1973.

[139] Z. Lin and S. Fu. On 1212-avoiding restricted growth functions. *The Electronic Journal of Combinatorics*, 24(1):P53, electronic, 2017.

[140] A. Lipowski and D. Lipowska. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 391(6):2193–2196, Mar 2012.

[141] Q. Liu, X. Li, L. Gao, and Y. Li. A modified genetic algorithm with new encoding and decoding methods for integrated process planning and scheduling problem. *IEEE Transactions on Cybernetics*, pages 1–10, 2020.

[142] M. Lothaire. *Combinatorics on words*. Addison-wesley, 1983.

[143] H. Maaranen, K. Miettinen, and M. Mäkelä. Quasi-random initial population for genetic algorithms. *Computers & Mathematics with Applications*, 47(12):1885–1895, 2004.

[144] P. MacMahon. The indices of permutations and derivation therefrom of functions of a single variable associated with the permutations of any assemblage of objects. *Amer. J. Math.*, 35:281–322, 1913.

[145] T. Mansour. Counting peaks at height k in a Dyck path. *Journal of Integer Sequences*, 5:Article 02.1.1, 03 2002.

[146] T. Mansour and M. Shattuck. Counting Dyck paths according to the maximum distance between peaks and valleys. *Journal of Integer Sequences*, 15:Article 12.1.1, 05 2012.

[147] T. Mansour and M. Shattuck. Some enumerative results related to ascent sequences. *Discrete Mathematics*, 315-316:29–41, 2014.

[148] T. Mansour and M. Shattuck. Pattern avoidance in inversion sequences. *Pure Mathematics and Applications*, 25(2):157–176, 2015.

[149] T. Mansour and V. Vajnovszki. Efficient generation of restricted growth words. *Information Processing Letters*, 113(17):613–616, 2013.

[150] T. Mansour and V. Vajnovszki. Efficient generation of restricted growth words. *American Discrete Mathematics*, page 113:613–616, 2013.

[151] R. Mantaci and F. Rakotondrajao. A permutations representation that knows what eulerian means. *Discrete Mathematics and Theoretical Computer Science*, 4, 12 2001.

[152] G. Mathews and P. MacMahon. Combinatory analysis. vol. i. *The Mathematical Gazette*, 8(118):125, 1915.

[153] M. Mehdi. *Paralell hybrid optimization methods for permutation based problems*. Thesis, Université des Sciences et Technologie de Lille - Lille I, Oct. 2011.

[154] D. Merlini, R. Sprugnoli, and C. Verri. Some statistics on Dyck paths. *Journal of Statistical Planning and Inference*, 101(1):211–227, 2002.

[155] H. Mohammed Ali, W. Abdou, P. Chatonnay, C. Bloch, and F. Spies. Behaviour study of an evolutionary design for permutation problems. *International Congress on Information and Communication Technology , London, UK*, 2018.

[156] H. Mohammed Ali, C. Bloch, W. Abdou, P. Chatonnay, and F. Spies. Behaviour Study of an Evolutionary Design for Permutation Problems. In *International Congress on Information and Communication Technology*, volume 797 of *Advances in Intelligent Systems and Computing*, pages 845 – 853, London, United Kingdom, Feb. 2018.

[157] G. Mohanty. Path counting—simple boundaries. *Lattice Path Counting and Applications*, page 1–29, 1979.

[158] S. Muelas, J. Peña, V. Robles, and A. LaTorre. Voronoi-initializated island models for solving real-coded deceptive problems. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, GECCO '08, page 993–1000, New York, NY, USA, 2008. Association for Computing Machinery.

[159] A. Nebro, J. Durillo, and M. Vergne. Redesigning the jMetal multi-objective optimization framework. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, page 1093–1100, New York, NY, USA, 2015. Association for Computing Machinery.

[160] D. Noever and S. Baskaran. Steady-state vs. generational genetic algorithms: A comparison of time complexity and convergence properties, 07 1992.

[161] R. Ohira, M. Islam, J. Jo, and B. Stantic. Amga: An adaptive and modular genetic algorithm for the traveling salesman problem. *Intelligent Systems Design and Applications*, 2020.

[162] I. Osman and G. Laporte. Metaheuristics: A bibliography. *Annals of Operational Research*, 63:513–628, 10 1996.

[163] R. Oste and J. Jeugt. Motzkin paths, Motzkin polynomials and recurrence relations. *The Electronic Journal of Combinatorics*, 22(2), 2015.

[164] S. Pesko. Differential evolution for small tsps with constraints. *Fourth International Scientific Conference : Challenges in Transport and Communications*, 2006.

[165] W. Pierskalla. The tri-substitution method for the three-dimensional assignment problem. *Journal of the Canadian Operational Research Society*, 5:71–81, 1967.

[166] W. Pierskalla. Letter to the editor–the multidimensional assignment problem. *Operations Research*, 16:422–431, 04 1968.

[167] V. Pillwein and C. Schneider. *Algorithmic Combinatorics: Enumerative Combinatorics, Special Functions and Computer Algebra In Honour of Peter Paule on his 60th Birthday: In Honour of Peter Paule on his 60th Birthday*. Springer, 01 2020.

[168] K. Plant and N. Stanton. The explanatory power of schema theory: theoretical foundations and future applications in ergonomics. *Ergonomics*, 56(1):1–15, 2013. PMID: 23140407.

[169] R. Poli. Schema theorems without expectations. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, GECCO'99, page 806, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[170] R. Poli. On fitness proportionate selection and the schema theorem in the presence of stochastic effects. 2000.

[171] R. Poli. Exact schema theory for genetic programming and variable-length genetic algorithms with one-point crossover. *Genetic Programming and Evolvable Machines*, 2(2):123–163, 2001.

[172] R. Poli. Recursive conditional schema theorem, convergence and population sizing in genetic algorithms. In W. N. Martin and W. M. Spears, editors, *Foundations of Genetic Algorithms 6*, pages 143–163. Morgan Kaufmann, San Francisco, 2001.

[173] R. Poli. Why the schema theorem is correct also in the presence of stochastic effects. 12 2001.

[174] R. Poli. Hyperschema theory for gp with one-point crossover, building blocks, and some new results in ga theory. 04 2002.

[175] R. Poli and W. Langdon. A review of theoretical and experimental results on schemata in genetic programming. pages 1–15, 01 1998.

[176] R. Poli and W. Langdon. Schema theory for genetic programming with one-point crossover and point mutation. *Evol. Comput.*, 6(3):231–252, Sept. 1998.

[177] R. Poli and N. McPhee. General schema theory for genetic programming with subtree-swapping crossover: Part i. *Evol. Comput.*, 11(1):53–66, Mar. 2003.

[178] R. Poli, N. Mcphee, and J. Rowe. Exact schema theory and markov chain models for genetic programming and variable-length genetic algorithms with homologous crossover. *Genetic Programming and Evolvable Machines*, 5:31–70, 03 2004.

[179] P. Poon and J. Carter. Genetic algorithm crossover operators for ordering applications. *Computers & Operations Research*, 22(1):135–147, 1995. Genetic Algorithms.

[180] A. Poore. Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking. *Computational Optimization and Applications*, 3:27–57, 03 1994.

[181] A. Poore and N. Rijavec. A lagrangian relaxation algorithm for multidimensional assignment problems arising from multitarget tracking. *Siam Journal on Optimization - SIAM J OPTIMIZATION*, 3, 08 1993.

[182] A. Poore, N. Rijavec, M. Liggins, and V. Vannicola. Data association problems posed as multidimensional assignment problems: problem formulation. In O. E. Drummond, editor, *Signal and Data Processing of Small Targets 1993*, volume 1954, pages 552 – 563. International Society for Optics and Photonics, SPIE, 1993.

[183] M. Portmann and A. Vignier. Performances' study on crossover operators keeping good schemata for some scheduling problems. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*, GECCO'00, page 331–338, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[184] V. Pratt. Computing permutations with double-ended queues, parallel stacks and parallel queues. *Proceedings of the fifth annual ACM symposium on Theory of computing - STOC 73*, 1973.

[185] L. Pudwell. Enumeration schemes for words avoiding permutations. *Permutation Patterns*, page 193–212, 2010.

[186] S. Rahnamayan, H. Tizhoosh, and M. Salama. A novel population initialization method for accelerating evolutionary algorithms. *Computers & Mathematics with Applications*, 53(10):1605–1614, 2007.

[187] D. Rogers and L. Shapiro. Some correspondences involving the Schröder numbers and relations. *Lecture Notes in Mathematics Combinatorial Mathematics*, page 267–274, 1978.

[188] P. Rosa, H. Sriwindono, A. Nugroho, A. Polina, and K. Pinaryanto. Comparison of crossover and mutation operators to solve teachers placement problem by using genetic algorithm. *Journal of Physics Conference Series*, 2020.

[189] S. Sahni and T. Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, July 1976.

[190] K. Sallam, S. Elsayed, R. Sarker, and D. Essam. *Differential Evolution with Landscape-Based Operator Selection for Solving Numerical Optimization Problems*, volume 8, pages 371–387. 11 2017.

[191] A. Schrijver. Combinatorial optimization, 2003.

[192] L. Shapiro. A Catalan triangle. *Discrete Mathematics*, 14(1):83–90, 1976.

[193] R. Simion and F. Schmidt. Restricted permutations. *European J. Combin*, page 383–406, 1985.

[194] R. Smith. Two stacks in series: A decreasing stack followed by an increasing stack. *Annals of Combinatorics*, 18:359–363, 2014.

[195] R. Smith and V. Vatter. A stack and a pop stack in series, 2013.

[196] N. Soni and T. Kumar. Study of various mutation operators in genetic algorithms. 2014.

[197] W. Spears and V. Anand. A study of crossover operators in genetic programming. *Proceeding of the Sixth International Symposium on Methodologies for Intelligent Systems*, 542, 07 1992.

[198] R. Stanley. What is enumerative combinatorics? *Enumerative Combinatorics*, page 1–63, 1986.

[199] R. Stanley. *Enumerative Combinatorics*, volume 2 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1999.

[200] R. Stanley. *Catalan Numbers*. Cambridge University Press, 2015.

[201] R. Sulanke. Bijective recurrences concerning Schröder paths. *The Electronic Journal of Combinatorics*, 5(1), 1998.

[202] R. Sulanke. Catalan path statistics having the Narayana distribution. *Discrete Mathematics*, 180(1):369–389, 1998. Proceedings of the 7th Conference on Formal Power Series and Algebraic Combinatorics.

[203] R. Sulanke. Constraint-sensitive Catalan path statistics having the Narayana distribution. *Discrete Mathematics*, 204(1):397–414, 1999. Selected papers in honor of Henry W. Gould.

[204] E. Talbi. *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009.

[205] K. Tan, S. Chiam, A. Mamun, and C. Goh. Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. *European Journal of Operational Research*, 197(2):701–713, 2009.

[206] P. Tang and M. Tseng. Adaptive directed mutation for real-coded genetic algorithms. *Applied Soft Computing*, 13:600–614, 01 2013.

[207] L. Tao and X. Xin. Study of artificial immune based schema theorem. *The Journal of Information and Computational Science*, 11(11):3663–3671, 2014.

[208] R. Tarjan. Sorting using networks of queues and stacks. *Journal of the ACM*, 19(2):341–346, 1972.

[209] P. Terwilliger. Using Catalan words and a q-shuffle algebra to describe a pbw basis for the positive part of $u_q(\widehat{\mathfrak{sl}}_2)$, 2018.

[210] A. Thue. *Uber unendliche Zeichenreihen*. Kristiania, 1906.

[211] A. Thue. *Uber die gegenseitige Lage gleicher Teile gewisser Zeichenreihen*. Kristiania, 1912.

[212] A. Thue. *Probleme uber Veranderungen von Zeichenreihen nach gegebenen Regeln*. Kristiania, 1914.

[213] TSPData. Tsplib. http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp.

[214] G. Ücoluk. Genetic algorithm solution of the tsp avoiding special crossover and mutation. *Intelligent Automation & Soft Computing*, 8(3):265–272, 2002.

[215] H. Ulfarsson. Describing West-3-stack-sortable permutations with permutation patterns, 2012.

[216] V. Vajnovszki. A loopless generation of bitstrings without p consecutive ones. In C. S. Calude, M. J. Dinneen, and S. Sburlan, editors, *Combinatorics, Computability and Logic*, pages 227–240, London, 2001. Springer London.

[217] A. Vie. Qualities, challenges and future of genetic algorithms: a literature review, 2021.

[218] M. Vlach. Branch and bound method for the three-index assignment problem. *Ekonomicko-Matematicky Obzor*, 3:181–191, 01 1967.

[219] J. von Neumann. Various techniques used in connection with random digits. In A. S. Householder, G. E. Forsythe, and H. H. Germond, editors, *Monte Carlo Method*, volume 12 of *National Bureau of Standards Applied Mathematics Series*, chapter 13, pages 36–38. US Government Printing Office, Washington, DC, 1951.

[220] A. Wadhwa. *Analysis of Selection Techniques in Genetic Algorithm*. 06 2016.

[221] J. West. *Permutations with forbidden subsequences, and, stack-sortable permutations*. Massachusetts Institute of Technology, 1990.

[222] J. West. Sorting twice through a stack. *Theoretical Computer Science*, 117(1):303–313, 1993.

[223] J. West. Generating trees and the Catalan and Schröder numbers. *Discrete Mathematics*, 146(1-3):247–262, 1995.

[224] J. West. Generating trees and forbidden subsequences. *Discrete Mathematics*, 157(1-3):271–283, 1996.

[225] D. White. An overview of schema theory, 2014.

[226] D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4, 10 1998.

[227] L. Whitley. Cellular genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, page 658, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[228] K. William. Proof of a conjectured q,t-Schröder identity, 2010.

[229] A. Wright. The exact schema theorem. *Computing Research Repository - CORR*, 05 2011.

[230] C. Yan and Z. Lin. Inversion sequences avoiding pairs of patterns. *Discrete Mathematics & Theoretical Computer Science*, vol. 22 no. 1, June 2020.

[231] X. Yu and M. Gen. *Introduction to Evolutionary Algorithms*. Springer Publishing Company, Incorporated, 2012.

[232] W. Yuping. Schema theorem based on limited character set encoding and its proof. *Journal of Xidian University*, 2012.

[233] D. Zeilberger. A proof of Julian Wests conjecture that the number of two-stack sortable permutations of length n is 2(3n)!/((n+1)!(2n+1)!). *Discrete Mathematics*, 102(1):85–93, 1992.

[234] Y. Zhou. Study on genetic algorithm improvement and application. Thesis, Worcester Polytechnic Institute, 100 Institute Road, Worcester MA 01609-2280 USA, May 2006.

[235] Z. Zojaji and M. Ebadzadeh. Semantic schema theory for genetic programming. *Applied Intelligence*, 44, 07 2015.

Appendix A

List of publications

Catalan words avoiding pairs of length three patterns

Jean-Luc Baril Carine Khalil Vincent Vajnovszki

LIB, Université de Bourgogne Franche-Comté, Dijon, France

received 25th Dec. 2019, revised 31st Mar. 2021, accepted 1st Apr. 2021.

Catalan words are particular growth-restricted words counted by the eponymous integer sequence. In this article we consider Catalan words avoiding a pair of patterns of length 3, pursuing the recent initiating work of the first and last authors and of S. Kirgizov where (among other things) the enumeration of Catalan words avoiding a patterns of length 3 is completed. More precisely, we explore systematically the structural properties of the sets of words under consideration and give enumerating results by constructive bijections or bivariate generating functions with respect to the length and descent number. Some of the obtained enumerating sequences are known, and thus the corresponding results establish new combinatorial interpretations for them.

Keywords: Catalan/pattern-avoiding words, enumeration, constructive bijections, (bivariate) generating functions.

1 Introduction and notation

Catalan words are particular growth-restricted words and they represent still another combinatorial class counted by the Catalan numbers, see for instance [12, exercise 6.19.u, p. 222]. This paper contributes to a recent line of research on classical pattern avoidance on words subject to some growth restrictions (for instance, ascent sequences [2, 6], inversion sequences [5, 10, 14], restricted growth functions [4, 9]) by investigating connections between sequences on the On-line Encyclopedia of Integer Sequences [11] and Catalan words avoiding two patterns of length 3.

Through this paper we consider words over the set of non-negative integers and we denote such words by sequences (for instance $w_1w_2\dots w_n$) or by italicized boldface letter (for instance $\boldsymbol{w}$ and $\boldsymbol{u}$). The word $\boldsymbol{w} = w_1w_2\dots w_n$ is called a *Catalan word* if

$$w_1 = 0 \text{ and } 0 \leq w_i \leq w_{i-1} + 1 \text{ for } i = 2, 3, \dots, n.$$

Catalan words are in bijection with maybe the most celebrated combinatorial class having the same enumerating sequence: Dyck paths[(i)]. Indeed, in a length $2n$ Dyck path collecting for the up steps the ordinates of their starting points we obtain a length n Catalan word, and this construction is a bijection. See Figure 1 where this bijection is depicted for an example. We denote by $\mathcal{C}_n$ the set of length n Catalan words and $c_n = |\mathcal{C}_n|$ is the nth Catalan number $\frac{1}{n+1}\binom{2n}{n}$.

A *pattern* is a word with the property that if i occurs in it, then so does j, for any j with $0 \leq j < i$. A pattern $\pi = \pi_1\pi_2\dots\pi_k$ is said to be contained in the word $\boldsymbol{w} = w_1w_2\dots w_n$, $k \leq n$, if there is a sub-word of $\boldsymbol{w}$, $w_{i_1}w_{i_2}\dots w_{i_k}$, order-isomorphic with $\pi_1\pi_2\dots\pi_k$. If $\boldsymbol{w}$ does not contain π, we say that $\boldsymbol{w}$ *avoids* π, see for instance Kitaev's seminal book [7] on this topic.

For a pattern π, we denote by $\mathcal{C}_n(\pi)$ the set of length n Catalan words avoiding π, and $c_n(\pi) = |\mathcal{C}_n(\pi)|$ is the cardinality of $\mathcal{C}_n(\pi)$ and $\mathcal{C}(\pi) = \cup_{n\geq 0}\mathcal{C}_n(\pi)$. For example, $\mathcal{C}_n(101)$ is the set of length n Catalan words avoiding 101, that is, the set of words $\boldsymbol{w}$ in $\mathcal{C}_n$ such that there are no i, j and k, $1 \leq i < j < k \leq n$, with $w_i = w_k > w_j$. So, $\mathcal{C}_4(101) = \{0000, 0001, 0010, 0011, 0012, 0100, 0110, 0111, 0112, 0120, 0121, 0122, 0123\}$. Likewise, if π is the set of patterns $\{\alpha, \beta, \dots\}$, then $\mathcal{C}_n(\pi)$ and $\mathcal{C}_n(\alpha, \beta, \dots)$ denote both the set of length n Catalan words avoiding each pattern in π; and $c_n(\pi) = c_n(\alpha, \beta, \dots)$ and $\mathcal{C}(\pi) =$

(i) A Dyck path is a path in the first quadrant of the plane which begins at the origin, ends at $(2n, 0)$, and consists of up steps $(1, 1)$ and down steps $(1, -1)$.

Information Processing Letters 171 (2021) 106138

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Catalan and Schröder permutations sortable by two restricted stacks

Jean-Luc Baril [a], Giulio Cerbai [b,1], Carine Khalil [a], Vincent Vajnovszki [a,*]

[a] LIB, Université de Bourgogne Franche-Comté, B.P. 47 870, 21078 Dijon Cedex, France
[b] Dipartimento di Matematica e Informatica "U. Dini", University of Firenze, Firenze, Italy

ARTICLE INFO

Article history:
Received 14 September 2020
Received in revised form 15 February 2021
Accepted 2 May 2021
Available online 19 May 2021
Communicated by Amit Chakrabarti

Keywords:
Stack sorting/two stacks in series
Pattern avoiding permutations/machines
Catalan and the Schröder numbers
Combinatorial problems

ABSTRACT

Pattern avoiding machines were introduced recently by Claesson, Cerbai and Ferrari as a particular case of the two-stacks in series sorting device. They consist of two restricted stacks in series, ruled by a right-greedy procedure and the stacks avoid some specified patterns. Some of the obtained results have been further generalized to Cayley permutations by Cerbai, specialized to particular patterns by Defant and Zheng, or considered in the context of functions over the symmetric group by Berlow. In this work we study pattern avoiding machines where the first stack avoids a pair of patterns of length 3 and investigate those pairs for which sortable permutations are counted by the (binomial transform of the) Catalan numbers and the Schröder numbers.

1. Introduction

Pattern avoiding machines were recently introduced in [7] in attempt to gain a better understanding of sortable permutations using stacks in series. They consist of two restricted stacks in series, equipped with a right-greedy procedure, where the first stack avoids a fixed pattern, reading the elements from top to bottom; and the second stack avoids the pattern 21 (which is a necessary condition for the machine to sort permutations). The authors of [7] provide a characterization of the avoided patterns for which sortable permutations do not form a class, and they show that those patterns are enumerated by the Catalan numbers. For specific patterns, such as 123 and the decreasing pattern of any length, a geometrical description of sortable permutations is also obtained. The pattern 132 has been solved later in [8]. Some of these results have been further generalized to Cayley permutations in [9]. More recently, Berlow [5] explores a single stack version of pattern avoiding machines, where the stack avoids a set of patterns and the sorting process is regarded as a function. Analogous machines, but based on the notion of consecutive patterns, have been introduced and discussed in [10].

In this work we study a variant of pattern-avoiding machines where the first stack avoids (σ, τ), a pair of patterns of length three. Following [7], we call it (σ, τ)-machine. More specifically, we restrict ourselves to those pairs of patterns for which sortable permutations are counted by either the Catalan numbers or two of their close relatives: the binomial transform of Catalan numbers and the Schröder numbers. For the pair $(132, 231)$ we show that sortable permutations are those avoiding 1324 and 2314, a set whose enumeration is given by the large Schröder numbers. Under certain conditions on the avoided patterns, the output of the first stack is bijectively related to its input (see [5,9]): it follows that for three pairs of patterns, namely $(123, 213)$, $(132, 312)$ and $(231, 321)$, sortable permutations are counted by the Cata-

* Corresponding author.
E-mail addresses: barjl@u-bourgogne.fr (J.-L. Baril), giulio.cerbai@unifi.it (G. Cerbai), carine.khalil@u-bourgogne.fr (C. Khalil), vvajnov@u-bourgogne.fr (V. Vajnovszki).

[1] G.C. is member of the INdAM Research group GNCS; he is partially supported by INdAM-GNCS 2020 project "Combinatoria delle permutazioni, delle parole e dei grafi: algoritmi e applicazioni".

https://doi.org/10.1016/j.ipl.2021.106138
0020-0190/

Transmission of Genetic Properties in Permutation Problems: Study of Lehmer Code and Inversion Table Encoding

Carine Khalil( ) and Wahabou Abdou

LIB, Université de Bourgogne Franche-Comté, B.P. 47 870, 21078 Dijon, France
{carine.khalil,wahabou.abdou}@u-bourgogne.fr

Abstract. Solution encoding describes the way decision variables are represented. In the case of permutation problems, the classical encoding should ensure that there are no duplicates. During crossover operations, repairs may be carried out to correct or avoid repetitions. The use of indirect encoding aims to define bijections between the classical permutation and a different representation of the decision variables. These encodings are not sensitive to duplicates. However, they lead to a loss of genetic properties during crossbreeding. This paper proposes a study of the impact of this loss both in the space of decision variables and in that of fitness values. We consider two indirect encoding: the Lehmer code and the Inversion table.

Keywords: Genetic algorithm · Permutation problems · TSP · Encoding · Lehmer code · Inversion table

1 Introduction

Permutation-based optimization problems are widely studied in the literature because of their hardness and the diversity of their application fields. They are particularly used in the domain of network device deployment, scheduling or transportation. Solving such problems consists of finding a permutation that minimizes/maximizes some criteria.

Many efficient methods exist for solving permutation problems. This paper focuses on Genetic Algorithms (GAs) which are powerful stochastic optimization techniques. They are inspired by Darwin's theory of evolution and natural selection. GAs help with the exploration of a search space in order to find an optimal or a near optimal solution for a given problem. In GAs, a possible solution to the optimization problem is referred to as an individual. Generally, the algorithm starts with a randomly generated set of individuals (population). This population evolves throughout generations towards good solutions. At each generation of the genetic process, each individual in the population is evaluated based on objective function(s). This leads to the computation of a fitness value

L. Rutkowski et al. (Eds.): ICAISC 2021, LNAI 12854, pp. 1–10, 2021.
https://doi.org/10.1007/978-3-030-87986-0_35