eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Doktorego Tesia
Matematika eta Estatistika

Algorithms for Large Orienteering Problems

Tamaina Handiko Orientazio Problementzako Algoritmoak

Gorka Kobeaga

Zuzendariak:
María Merino
Jose A. Lozano

2021ko Urtarrila

PhD Thesis
Mathematics & Statistics

Algorithms for Large Orienteering Problems

Gorka Kobeaga

Advisors:
María Merino
Jose A. Lozano

January 2021

This research was carried out at the Basque Center for Applied Mathematics (BCAM) within the Machine Learning Group. This research was supported by the Basque Government through the BERC (2018-2021, 2014-2017) and ELKARTEK programs, and by the Spanish Goverment through the grant BES-2015-072036, the BCAM Severo Ochoa accreditations SEV-2013-0323 and SEV-2017-0718, and the projects PID2019-104933GB-I00 and MTM2015-65317-P.

Tesiaren nondik norakoak

Tesi lan honetan, tamaina handiko Orientazio Problemak (OP) ebazteko algoritmoak garatu ditugu. OP optimizazio konbinatorioko problema bat da: herri multzo bat eta hauen arteko distantzia emanik, herri bakoitzak bere saria duelarik, eta ibilbidearen distantzia (edo denbora) osoaren murrizketa bat ezarririk, OPren helburua sarien batura maximizatzen duen ibilbidea aurkitzean datza.

Problema honek, optimizazio konbinatorioko bi problema klasikorekin lotura estua du, izan ere, Saltzaile Ibiltariaren Problemaren (TSP) eta Motxilaren Problemaren (KP) arteko konbinazio bezala ikusi daiteke. Batetik, TSPren helburua herri multzo bat eta hauen arteko distantzia emanik, herri guztiak behin bakarrik bisitatzen duen ibilbide laburrena aurkitzean datza. Bestetik, KPn objektu multzo bat emanik, bakoitzak bere saria eta pisua duelarik, eta motxilak izan dezakeen gehienezko pisua ezarririk, helburua motxilan sartzen den eta sarien batura maximizatzen duten objektu azpimultzoa aukeratzean datza.

Problemaren izenak orientazio lasterketa bezala ezagutzen den kirol batean du jatorria. Kirol honetako parte-hartzaileei mapa topografiko bat ematen zaie, kontrol gune batzuk zehaztuta dituena eta helburua, denbora tarte batean, ahalik eta kontrol gune gehienetatik pasatzea da. Lasterketaren hasierako eta bukaerako kontrol guneak aurretiaz zehaztuta egoten dira, eta emandako denbora tartearen barruan bukaerara iristen ez diren parte-hartzaileak jokoz kanpo gelditzen dira. Aldaerak aldaera, lasterketa modalitatearen arabera, kontrol guneek puntuazio desberdinak izan ditzakete.

Problemaren izenaren jatorria kirol bat baden arren, OPk aplikazio ugari ditu. Esate baterako, lanaldi batean herri (saltoki) guztiak bisitatzeko denbora ez duen saltzaileak, bere lehentasunen arabera, lanaldirako ibilbide aproposena aukeratu behar du, eta funtsean hori da OPren bidez ebazten dena. OP eta bere aldaerak aztertzen dituzten lanek izan duten azken urteetako gorakada, problema hauek turismo bidaien plangintzan duten erabileran dago oinarritua.

Hiri bat bisitatzera doan turistarentzat, ohikoa izaten da bisitaren luzapen-denboraren mugarengatik, hirian aukeran dauden jarduera, ikuskizun eta gune guztiez gozatzeko aukera ez izatea. Horrela, bisitariak eskuragai dauden jarduera guztietatik batzuk bakarrik bisitatu ahal izango ditu. Bisita planifikatzeko, jarduera bakoitzari lehentasun bat ezarri behar izaten da, eta bisita ahalik eta gustukoena izateko, lehentasun hauek maximizatzen duen ibilbidea aurkitu behar da. Horretarako, jardueren lehentasunez eta duten denboraz gain, bidaiariek kontutan izan behar izaten dute jardueren arteko distantzia eta ostatatuta dauden hotela. Errealitatean ibilbide gustukoena aukeratzearen problema konplexuagoa da (jardueren arteko denborak ez dira momentu oro berdinak, jarduera batzuk ez daude eguneko 24 orduetan zabalik, lehentasunak aurreiritziak dira, egun bat baino gehiago izan ditzake bisita egiteko) eta OPren aldaerek konplexutasun horri erantzuna ematen saiatzen dira. Hala ere, tesi honetan OPren bertsio klasikoa aztertzen dugu, eta helburua ahalik eta tamaina handieneko problemak aztertzeko teknika eta algoritmoak garatzea izan da.

OP problema hurrengo eran formulatu daiteke era sinple batean:

$$\begin{aligned} \max \quad & \tau \text{ ibilbideak bisitatutako herrien sarien batura} \\ \text{h.b.} \quad & \tau \text{ ibilbidea ziklo sinplea da,} \\ & \tau \text{ ibilbidearen luzera ez da } d_0 \text{ baina handiagoa,} \\ & \tau \text{ ibilbideak 1 herria bisitatzen du} \end{aligned}$$

non d_0 zikloaren gehienezko luzera eta 1 hasierako herria (hotela) diren. Definitzeko erreza den problema hau, praktikan, ebaztea zaila da. *NP-hard* problema bat da, izan ere, ibilbide Hamiltondarra aurkitzearen *NP-complete* problema klasikoa, OPren kasu partikularra da. Honez gain, herri multzo bat OPren soluzio bideragarriren baten parte den zehaztea ere problema zaila da. Hau da, herri multzo bat emanik, herri guzti hauetatik igarotzen den d_0 baino luzera txikiagodun ibilbiderik existitzen baden *NP-complete* problema bat da, hau TSPren erabakitze bertsioa baita.

OP ebazteko, algoritmo heuristiko bat eta algoritmo zehatz bat garatu ditugu. Aldi berean, ziklo problementzako algoritmo zehatzaren parte diren euskarri grafoen sinplifikazio teknika eta azpizikloak identifikatzeko separazio algoritmoak. Izan ere, teknika hauek, OP problemaz gain, soluzioa ziklo sinple bat duten edozein problema ebazteko erabilgarriak dira.

Lanaren 2. kapituluan, EA4OP izena eman diogun, OPrentzako algoritmo metaheurisitko bat aurkeztu dugu. Zehazki, EA4OP algoritmo ebolutibo bat da, hau da, ibilbideen populazioa sortzen du eta populazio hau eboluzionatzen du populazioko soluzioen kalitatea hobetze aldera.

Hasierako soluzioak sortzeko, lehenengo, soluzioan egongo diren herriak aukeratzen ditugu Bernoulli banaketaren bidez, eta gero, herri horietatik pasatzen den ibilbidea eraikitzen dugu. Hasierako herriak aukeratzeko, herri multzo osoaren TSP soluzio hurbildua aurkitzen dugu eta TSParen soluzioaren balioaren, $v(TSP)$, eta OPren distantzia murrizketaren arteko erlazioaz baliatuz, herri bat hasierako soluzioan egoteko probabilitatea zehazten dugu, $p = \sqrt{d_0/v(TSP)}$.

EA4OPren ezaugarri nagusienetako bat, algoritmo azkar bat izateko xedez, soluzio ez bideragarriekin lan egitea da. Hori dela eta, algoritmoaren garapenean, hasiera faseaz gain, bi fase bereizten dira: eboluzio fasea eta soluzio bideragarriak berreskuratzeko fasea. Belaunaldiz belaunaldi gauzatzen den eboluzio faseak hiru eragile barnebiltzen ditu: gurasoen aukeraketa, gurutzaketa eta mutazioa. Eboluzio fasean, populazioko soluzioak ez-bideragarriak izan litezkeenez, belaunaldi kopuru baten ostean soluzio bideragarriak berreskuratzen ditugu populazioko soluzioak moldatuz (`ken` eragilea), eta ostean bilaketa lokal bat (`gehi` eragilea) aplikatzen diegu soluzio berri hauei.

Eragile genetikoen ikuspuntutik lan honen ekarpen nagusia OPrentzako, eta orokorrean ziklo problementzako, gurutzaketa eragile berri bat garatzea izan da. Eragile hau, TSPrentzako proposatutako Ertzen Birkonbinazio Gurutzaketan (Edge Recombination Crossover, Whitley et al. [1989]) oinarrituz orokortu dugu. Ziklo problementzako gurutzaketa eragile berri honek bi aldetan jartzen du fokua: batetik, soluzio gurasoetan bisitatzen diren erpin komunak, soluzio umean ere bisitatzea, eta bestetik, soluzio umearen ibilbidean, soluzio gurasoetan erabiltzen diren ertzek lehentasuna izatea.

EA4OPren beste ekarpen bat problema handiak ebazteko aproposa den bilaketa lokala da. OPn bilaketa lokal eraginkor eta erabiliena, ibilbidean ez dauden herriak ibilbidera sartzeko (distantzia osoa kontuan izanik) prozedura da, baina hau oso astuna da. Izan ere, kanpoko herri bakoitzerako, behin eta berriz, ibilbidean sartzeko posizio hoberena aurkitu behar da. Lan honetan, kanpoko herriak ibilbidean sartzeko aukerak murrizten ditugu. Horretarako, k-d zuhaitzak erabiltzen ditugu, kanpoko herri bakoitzeko ibilbidean dauden hiru herri hurbilenak bilatzeko, eta hauen ondoz-ondoan txertatzeko aukera bakarrik hartzen dugu kontutan.

Esperimentuek erakusten dute, EA4OP algoritmoak literatuko algoritmoen heuristikoek baino emaitzak hobeagoak lortzen dituela. Tamaina ertaineko problemetan (400 herri baino gutxiago), beste algoritmoekin konparatuz, EA4OP algoritmo lehiakorra dela ikusi dugu,. Aldiz, EA4OP nagusitasuna argi gelditzen da tamaina handiko problemetan (7393 herri arte), instantzia gehienetan algoritmo aurkariak baino emaitza hobeak eta arinagoak lortuz.

Tesiaren 3. kapitulak eta 4. kapituluak OPrentzako algoritmo zehatza dute aztergai. OPren soluzioak zikloak direnez, 3. kapituluan ziklo problementzako *Branch-and-Cut* algoritmoen parte diren prozedura komunak aztertzen ditugu. 4. kapituluan, OPrentzako *Branch-and-Cut* algoritmoa garatu eta honen emaitza konputazionalak konparatzen ditugu.

Izan bitez $G = (V, E)$, V erpinak eta E ertzak dituen grafo ez-zuzendua; $\mathcal{C}_G$, G grafoko ziklo sinpleen multzoa; eta $\mathbb{R}^V$ eta $\mathbb{R}^E$, V eta E bidez indexatutako bektore errealak. Izan bedi $(y, x)^\tau$, τ zikloaren bektore karakteristikoa, non $y_v = 1$ edo $x_e = 1$ baldin eta v erpina edo e ertza, hurrenez hurren, zikloan bisitatuta badaude. G grafoaren Ziklo Politopoa, P_C^G, G grafoko ziklo sinpleen bektore karakteristikoen inguratzaile konbexua da, hau da, $P_C^G := conv\{(y, x)^\tau \in \mathbb{R}^{V \times E} : \tau \in \mathcal{C}_G\}$. Ziklo problemen, eta bereziki OPren, soluzioak P_C^G politopoaren erpinak dira. Branch-and-Cut algoritmoek, problemaren optimoa lortze aldera, P_C^G espazioa (edo problemari dagokion soluzio espazio) modu eraginkor eta ordenatu batean arakatzea ahalbidetzen dute. Baina aurretiaz, P_C^G

espazioa (konbexua) murrizketa linealen bidez (hiperplanoen ebakidura bezala) adierazi behar da:

$$\begin{aligned} x(\delta(v)) - 2y_v &= 0, && v \in V \\ y_v - x_e &\geq 0, && \forall v \in V,\ e \in \delta(v) \\ x(\delta(Q)) - 2y_v - 2y_w &\geq -2, && v \in Q \subset V, 3 \leq |Q| \leq |V| - 3,\ w \in V - Q \\ x(E) &\geq 3, && \\ 1 \geq y_v &\geq 0, && \forall v \in V \\ x_e &\geq 0, && \forall e \in E \\ x_e &\in \mathbb{Z} && \forall e \in E \end{aligned}$$

non $\delta(Q)$ multzoa Q multzoren muga zeharkatzen duten ertzek osatzen duten.

Ziklo problema baten optimoaren bilaketa egiteko, *Branch-and-Cut* algoritmoek, P_C^G adierazpenaren azken baldintza (aldagai osoena) erlaxatzen dute, eta optimizatu ostean, balio ez osodun aldagairik izatekotan, bi azpi problemetan banatzen dute problema (*branching*). Honez gain, beste bi aspektu daude kontutan izan beharrekoak: (1) politopoaren erlaxazio linealarekin lan egiterakoan, bilaketa espazioa handitzen da eta, ondorioz, ebaketa gehigarriak erabiltzea komeni da, eta (2) P_C^G politopoaren adierazpenean dagoen bigarren murrizketa familiak (azpizikloak ezabatzeko murrizketak, SEC) kopuru esponentziala du. Bi aspektu hauek kontutan izanda, eta eraginkortasunari begira, *Branch-and-Cut* algoritmoa, politopoaren adierazpen sinplifikatu batekin abiarazten da (SEC murrizketa familia esponentziala kenduta) eta algoritmoan zehar, behar den heinean, ebaketa plano berriak (azpizikloak ezabatzeko murrizketak eta murrizketa gehigarriak) gehitzen dira.

Ebaketa plano egokiak bilatzeko, algoritmoan zehar, behin eta berriz, problema erlaxatuen soluzioekin lotutako euskarri grafoetan, G^*, separazio problema bezala ezagutzen direnak ebatzi behar dira. Separazio problema hauek ebaztea oso astuna da eta 3. kapituluan, ziklo problemen separazio algoritmoak arintzeko, uzkurtze teknika garatu dugu. Izan bitez G grafoa eta $S \subset V$ azpimultzoa, orduan $G[S]$ grafoa G grafoaren uzkurketa bat dela esaten da S multzoko erpin guztiak bakarra izango balira bezala kontsideratzen badira.

Hala ere, ebaketak galdu daitezkeenez, edozein uzkurketak ez du balio separazio problemen aurreprozesu bezala. Azpimultzo uzkurgarriak aurkitzeko, hiru erregela seguru (C1, C2 eta C3) orokortu ditugu P_C^G politopoarentzat. Behin ziklo politoporako baliogarria den murrizketa familia bat zehaztuta, uzkurketa erregela zorrotzagoak garatu daitezke murrizketa familia zehatz horrentzat. 3. kapituluan, SEC murrizketentzako uzkurtze bi erregela berezi (S1 eta S2) aurkezten ditugu.

Uzkurketa tekniken eragina, SEC murrizketen banatze problemetan neurtu dugu. Horretarako, lehenengo SEC murrizketen banatze algoritmoak aztertu ditugu, TSPtik orokortutako bi banatze algoritmo zehatz aurkeztuz. Esperimentuetan ikusi dugu uzkurtze teknikek, bereziki S1 eta S2 erregelen konbinazioak, 50 aldiz azkartu dezaketela SEC murrizketen banaketa algoritmoa eta beraz oso eraginkorrak eta aproposak direla ziklo problemen *Branch-and-Cut* algoritmoentzako. Gainera, uzkurtze teknikez gain, banantze algoritmoak arintzeko teknika konkretuak (S3 erregela) erabiliz, banantze algortimo hauek 250 aldiz azkartu daitezkeela ikusi dugu.

4. kapituluan *Branch-and-Cut* algoritmo bat garatu dugu OPrentzat. Algoritmo zehatz honek, literaturan aurretiaz argitaratutako lanak kontutan izateaz gain, hainbat ekarpen barnebiltzen ditu, eta hori dela eta OPrentzako Birjorratutako *Branch-and-Cut* (RB&C) algoritmoa izendatu dugu. Kontutan izan behar da OPrentzako azken algoritmo zehatza (Fischetti et al. [1998]) duela bi hamarkada baino gehiago argitaratu zela. Gure motibazioa TSP probleman erabilitako zenbait teknika OPrentzako orokortzea izan da.

Honako ekarpen hauek ditu 4. kapituluan aurkeztutako gure algoritmo zehatzak. Uzkurketa teknika darabilen, SEC eta Konektibitate Murrizketentzako (Connectivity Constraints, CC) banaketa algoritmo bat proposatu dugu. Aurreko kapituluan ikusitako uzkurketa teknikek eragin negatiboa dute CCn bilaketan, kontuan izan murrizketa hauek orokorrean ez direla ziklo politoporako baliogarriak. Duten eragin negatibo hori gutxitzeko asmoz, hiru prozedura proposatu ditugu CC gehigarriak bilatzeko.

Ziklo Politopoarako murrizketa gehigarri ezagunenak Blossom desberdintzak dira, TSPtik orokortuak Bauer [1997] lanean. Blossom murrizketentzako bi banaketa algoritmo heuristiko orokortu ditugu TSPn erabilitako Padberg-Hong (Padberg and Hong [1980]) eta Grötschel-Holland (Grötschel and Holland [1991]) algoritmoetan oinarrituz. Esperimentalki ikusi dugu, proposatutako bi heuristikek, literaturan ziklo problementzako blossom murrizketeten emaitzak hobetzen dituztela, bai soluzio kalitateri dagokionez baita algoritmoaren exekuzio denborari dagokionez ere.

RB&C algoritmoak Zutabe Sorrera (*Column Generation*) teknika darabil, honela bere LP azpiproblemetan aldagaien azpimultzo bat bakarrik erabiltzen du. Ondorioz, algoritmoaren urrats batzuetan baztertutako aldagaiak baloratu (*pricing*) behar dira, LP azpiproblemara sartu behar ote diren erabakitzeko. Baztertutako aldagai bakoitza baloratzeko, aldagaia parte den murrizketa guztiak hartu behar dira kontuan, eta hau baztertutako aldagai guztientzako kalkulatzea oso garestia da. Baloratze prozedura arintzeko, TSPrantzako Applegate et al. [2007] lanean proposatutako aldagaien baloratze teknikan oinarritu gara, honela kalkulu errepikakorrak behin bakarrik egitea lortzen dugu, eta garrantzitsuagoa dena, baztertutako aldagai gehienak zehazki baloratzea saihesten dugu.

RB&C algoritmoaren beste ekarpen bat banantze begizta hiru azpi-begiztatan banantzea da. Lehenengo begiztan banantze algoritmo arinak sartu ditugu. Bigarren begiztan OPren ziklo izaerarekin lotutako banantze algoritmoak. Hirugarren eta azken begiztan gainontzeko banantze algoritmoak. Esperimentalki ikusi dugu, banantze begizta hiru azpi-begiztetan banatzeak RB&C algoritmoa azkartzen duela.

OP problemen kalitatezko behe-mugak azkar lortzeko helburuz, RB&C algoritmo zehatzaren barnean, bi algoritmo heuristiko primal (primal heuristic) erabili ditugu. Lehengoak, ertz aldagaien balio primalak erabiltzen ditu. Bigarrenak, berriz, erpin aldagaien balio primaletan oinarrituz soluzio heuristikoen populazio bat sortzen du eta ostean populazioa eboluzionatzen du 2. kapituluan aurkeztutako EA4OP algoritmoa erabiliz. Lehenengo heuristika, bietan azkarrena, banantze begiztan erabiltzen da eta bigarrena, bietan kalitate onena lortzen duena, adarkatzeen ostean. OP problemen goi-mugak eguneratzeko kalkulua ere aurkeztu dugu.

Proposatutako RB&C algoritmoak primerako emaitzak lortu ditu egindako esperimentuetan. Konparatutako OPren 258 instantzietatik 180tan lortutako soluzioa optimoa dela egiaztatzen du, horietatik 18 lehenengo aldiz egiaztatu direlarik. Instantzia horietatik 245tan balio ezagun onena lortzen du, horietatik 76 balio berriak direlarik. Eta, 249 instantzian goi kota ezagun onena lortzen du, horietatik 85 berriak direlarik. Horrez gain, literaturako beste algoritmoekin buruz-buruko konparaketak egin ditugu soluzioaren kalitatea eta algoritmoaren exekuzio denbora konparatuz. RB&C algoritmoak tamaina ertaineko instantzietan emaitza lehiakorrak lortzen ditu, eta tamaina handiko problemetan berriz, emaitzarik onenak lortzen ditu.

OP ebazteko softwarearen garapena tesi honen zati garrantzitsu bat izan da. Hori dela eta, 5. kapituluan EA4OP eta RB&C algoritmoak instalatzeko eta erabiltzeko pausuak azaltzen ditugu.

Laburbiltzeko, tesi lan honetan tamaina handiko OPren instantziak ebazteko algoritmoak proposatu ditugu, heuristiko bat eta algoritmo zehatz bat, eta bi algoritmoek primerako emaitzak lortzen dituztela ikusi dugu, bai soluzioen kalitatearen aldetik eta bai azkartasunaren aldetik ere.

Thesis Summary

In this thesis, we have developed algorithms to solve large-scale Orienteering Problems (OP). OP is a combinatorial optimization problem, where given a weighted complete graph with vertex profits and a constant d_0, the goal is to find the simple cycle which, with a length lower than or equal to d_0, maximizes the sum of the profits of the visited vertices.

The OP can be seen as a combination of two classical combinatorial optimization problems: the Travelling Salesperson Problem (TSP) and the Knapsack Problem (KP). On the one hand, the purpose of the TSP is to find the shortest tour that visits each vertex exactly once. On the other hand, in the KP, given a set of objects each having its own reward and weight, and maximum weight of the knapsack, the problem consists of finding the subset of items that fits in the knapsack and maximizes the sum of the rewards.

The name of the problem originates from a sports game called orienteering. The participants are given a topographical map with detailed checkpoints, each with an associated score, and a time limit. The participants who visit the checkpoints that maximize the total obtained score within the time limit are the winners of the game.

Although the name of the problem originates from a sport, OP has a wide variety of applications. For example, a travelling salesperson without enough time to visit all the cities during a period of work must choose, according to their preferences, the most suitable route, and this is essentially what is decided by OP. The rise in recent years of works studying the OP and its variants is probably based on the applicability of the problem in tourism travel planning.

Commonly, a tourist visiting a city does not have time to enjoy all the activities and places in the city. In order to plan a visit as satisfactory as possible, the tourist must set a priority for each activity and find the tour that maximizes these priorities. For this purpose, in addition to the preferences of activities, the traveler must take into account the distance between the activities and lodging hotel. In reality, the problem of choosing the favorite tour is more complex (the time between activities is not the same at every moment, some activities are not open 24 hours a day, the preferences are preconceptions, the visit might last multiple days, etc.) and the variants of OP try to answer that complexity. However, in this thesis, we study the classical version of OP, and the goal has been to develop techniques and algorithms to solve problems as large as possible.

The OP problem can be formulated in the following simple way:

$$\begin{aligned} \max \quad & \text{total score of the vertices visited by } \tau && (0.3a) \\ \text{s.t.} \quad & \tau \text{ is a simple cycle,} && (0.3b) \\ & \tau \text{ has a length not greater than } d_0, && (0.3c) \\ & \tau \text{ visits the depot vertex} && (0.3d) \end{aligned}$$

where d_0 is the maximum length of the cycle. This problem, which is easy to define, is difficult to solve in practice. It is an NP-hard problem since the classical problem of finding a Hamiltonian tour is a particular case of the OP. Moreover, it is also difficult to determine whether a subset of vertices is part of any feasible solution of the OP. That is to say, given a subset of vertices, it is an NP-complete problem to determine if there exists a cycle with a length lower than d_0, since this is the decision version of the TSP.

To solve the OP, we have developed a heuristic algorithm and an exact algorithm. At the same time, and as part of the development of the exact algorithm for OP, we have generalized for cycle problems the support graph shrinking techniques and procedures to speed up the separation algorithms for subcycle elimination constraints developed for the TSP. These techniques, beyond the OP problem, are useful in solving any problem in which the solution is a simple cycle.

In Chapter 2, we have introduced the so-called EA4OP metaheuristic algorithm for OP. The EA4OP is an evolutionary algorithm, i.e., the algorithm creates a population of cycle solutions and evolves it to improve the quality of solutions in the population. To generate the initial solutions, we first choose the vertices that will be in each solution using the Bernoulli distribution, and then we build the route that passes through them. To select the initial vertices, we find an approximate TSP solution for the whole set of cities, and by using the relation between the value of the TSP solution, $v(TSP)$, and the distance constraint of the OP, we specify the probability ($p = \sqrt{d_0/v(TSP)}$) of including each city in the initial solution.

One of the key characteristics of EA4OP is to work with unfeasible solutions. Hence, in the development of the algorithm, apart from the beginning phase, there are two phases: the evolutionary phase and the feasible solution recovery phase. The evolutionary phase carried out from generation to generation, involves three operators: parent selection, crossover, and mutation. In the EA4OP algorithm, we recover the feasible solutions after a number of generations (the $d2d$ parameter) first by improving the route length and then by modifying the solutions in the population (drop operator). Once the solutions in the population are feasible we apply a local search (add operator) to these new solutions.

From the point of view of genetic operators, the main contribution of this work has been the development of a new crossover for OP, which in a wider context is also valid for any cycle problem. We have generalized this operator based on the Edge Recombination Crossover proposed for TSP (Whitley et al. [1989]). We are interested in inheriting two main characteristics from the parents related to the vertices and the edges. Regarding the visited vertices, the crossover maintains all the vertices that are common to both parent solutions, including, with some probability, the vertices that belong to only one parent, and excluding the vertices that do not belong to any parent solution. Regarding the route length, the crossover uses as many edges of the parents as possible in order to pass on the maximum amount of information and decrease length quality losses in the new child solution.

Another contribution in the EA4OP is the developed local search to handle large problems. The most widely used local search in OP is the procedure of introducing non-visited vertices to the route, but this is a very time-consuming procedure, since for every non-visited vertex, one must find the cheapest insertion position in the route. In this work, we reduce the possible insertion positions, for this purpose we use k-d trees, to search for each non-visited vertices the three nearest vertices in the route, and we only consider the possibility of inserting the non-visited vertex next to the three nearest ones in the route.

The experiments show that the EA4OP algorithm improves the results of the state-of-the-art heuristics. In medium-sized problems (fewer than 400 vertices) we found that EA4OP is a competitive algorithm obtaining similar results of the literature approaches. However, the superiority of the EA4OP is clearly seen for large-sized problems (up to 7393 vertices), where in most of the cases the EA4OP obtains better quality solutions in shorter execution times than competitor algorithms.

In chapters 3 and 4 we study exact algorithms for the OP. As the solutions of OP are cycles, in Chapter 3 we analyze the common procedures that are part of the Branch-and-Cut algorithms for cycle problems. In Chapter 4, we develop a specific Branch-and-Cut algorithm for OP and compare the computational results with the approaches in the literature.

Let $G = (V, E)$ be an undirected graph with no loops and denote by $\mathcal{C}_G$ the set of simple cycles in the graph G, and by $\mathbb{R}^V$ and $\mathbb{R}^E$ the space of real vectors whose components are indexed by elements of V and E, respectively. Then, the cycle polytope P_C^G of the graph G is the convex hull of the characteristic vectors of all the cycles of the graph, that is to say, $P_C^G := conv\{(y,x)^\tau \in \mathbb{R}^{V \times E} : \tau \in \mathcal{C}_G\}$. The solutions of the cycle problem, and particularly of the OP, are the vertices of the P_C^G. The Branch-and-Cut algorithms provide an efficient and orderly way to search the P_C^G space. In order to use the B&C approach, the polytope P_C^G must be characterized by means of a system of linear constraints:

$$
\begin{aligned}
x(\delta(v)) - 2y_v &= 0, && v \in V\\
y_v - x_e &\geq 0, && \forall v \in V,\ e \in \delta(v)\\
x(\delta(Q)) - 2y_v - 2y_w &\geq -2, && v \in Q \subset V, 3 \leq |Q| \leq |V| - 3,\ w \in V - Q\\
x(E) &\geq 3, &&\\
1 \geq y_v &\geq 0, && \forall v \in V\\
x_e &\geq 0, && \forall e \in E\\
x_e &\in \mathbb{Z} && \forall e \in E
\end{aligned}
$$

where $\delta(Q)$ is the set of edges in the coboundary of Q.

For the purpose of searching the optimal solution of a cycle problem, Branch-and-Cut algorithms relax the last constraint family in the expression of P_C^G (the integrality constraints), and after optimizing the relaxed system, in case the solution has non-integer values, it divides the problem into two subproblems (branching). There are two aspects to take into consideration: (1) when working with the linear relaxation of the polytope, the search space gets bigger, and therefore additional valid cuts are needed in order to explore efficiently the problem space, and (2) the second constraint family in the expression of the polytope P_C^G, the so-called Subcycle Elimination Constraints (SEC), has an exponential amount of constraints. Thus, the Branch-and-Cut algorithm starts with a simplified expression of the polytope (by excluding the SEC constraint family) and adds, when required, new cutting-planes (SECs and additional valid constraints) throughout the algorithm.

In order to find the appropriate cuts to add, it is needed to repeatedly solve the separation problems in the graphs associated with the solutions of the subproblems. In Chapter 3, we have developed the shrinking technique to speed up the algorithms to solve these separation problems. Given a graph G and a subset S of vertices, we denote by $G[S] = (V[S], E[S])$ the graph obtained by shrinking the set S into a single vertex.

However, since violated cuts might vanish with an arbitrary shrinking, not all the subsets are safe to shrink. Based on the definition given in [Padberg and Rinaldi, 1990b] for safe shrinking for the P_{TSP}^G, an analog definition can be formulated for safe shrinking for the P_C^G. We have obtained three safe shrinking rules (C1, C2, and C3) for the valid inequalities of the cycle polytopes. Depending on the inequality, more aggressive contractions can be employed as a preprocess of separation algorithms. In Chapter 3, we have also obtained two special shrinking rules (S1 and S2) for SECs.

We measure the impact of shrinking techniques on SEC separation problems. In the experiments, we have found that the shrinking techniques, in particular the combination of S1 and S2 rules can speed up the SEC reduction algorithm by 50 times, and are therefore very efficient and convenient for the Branch-and-Cut algorithms for cycle problems. We have also seen that using separation algorithm specific acceleration techniques (rule S3), in addition to shrinking, the speed up of the separation could be boosted 250 times.

In Chapter 4 we develop a Branch-and-Cut algorithm for the OP. This proposed algorithm, in addition to considering the previously published works in literature, brings multiple contributions together, hence the name of revisited Branch-and-Cut (RB&C) for OP. It must be noted that the last exact algorithm for the classical OP was published more than two decades ago (Fischetti et al. [1998]). Our motivation has been to generalize some of the succsesful techniques used in the TSP to OP.

We have proposed a joint separation algorithm for SECs and Connectivity Constraint (CC), which efficiently uses the shrinking technique by reducing the adverse effects of the shrinking for CCs.

The best known additional valid inequalities for the polytope cycle are the Blossom inequalities, generalized from TSP in Bauer [1997]. We have generalized two heuristic separation algorithms for blossoms based on the algorithms given by Padberg-Hong (Padberg and Hong [1980]) and Grötschel-Holland (Grötschel and Holland [1991]) for

TSP. Experimentally, we have seen that the two proposed heuristics improve the results of blossom separation heuristics in literature, both in terms of solution quality and in regard to the execution time of the RB&C algorithm.

During the B&C algorithm, only a subset of edges is included in the working linear relaxation. At certain points of the algorithm, we need to price the excluded edge variables, and add to the working problem: 1) to guarantee that the working relaxation is an upper bound of the problem or branched subproblem and 2) to recover, whenever it is possible, a feasible problem after feasibility breaking cuts have been added. Taking into account that usually only a small subset of variables is included in the relaxation, and that the excluded variables could participate in multiple cuts, the pricing phase could constitute a bottleneck in the B&C algorithm. We have developed a technique, inspired by that used in Applegate et al. [2007], which enables us to avoid repetitive calculations and to skip the exact calculation of the reduced cost of some variables.

Another contribution of the RB&C is the proposed separation loop for the OP that takes into consideration the different contributions and separation costs of the valid inequalities. The separation loop to find the violated cuts is accomplished in three subloops. In the inner loop, we consider two basic, but fast, separation algorithms. In the middle loop, we consider the separations of cuts which are related to the cycle essence of the OP. In the outer loop, we consider the rest of the cuts.

With the goal of obtaining good lower-bounds for the OP problems, we have used two primal heuristic algorithms: one heuristic uses the primal edge values, and the other heuristic uses the primal vertex values. Moreover, the second generates a population of heuristic solutions based on primal vertex values and then evolves the population using the EA4OP algorithm presented in Chapter 2. The first heuristic, the quickest of the two, is used in the separation loop, and the second heuristic, which attains the best quality solutions of the two, is used at the beginning of a branch node. We have also presented a calculation to update the upper-bounds during the branching phase.

The experiments have shown that the RB&C algorithm for OP is a much more efficient approach than the state-of-the-art B&C algorithm. It finds the optimality certification of the solutions in 180 out of 258 instances, from which 18 are new. Of these benchmark instances, in 245 best-known solution value is obtained, from which 76 are new values. And, in 249 instances, it obtains the best-known upper-bound values, from which 85 are new. In addition, we have made one-by-one comparisons with other algorithms in the literature comparing the quality of the solution and the execution time of the algorithm. In the case of medium-sized instances, the RB&C is able to obtain competitive results, while in the case of large-sized instances, it achieves the best results.

The development of OP software has been an important part of this thesis. In Chapter 5, we explain the steps to install and use the EA4OP and RB&C algorithms.

To summarize this thesis, we have proposed algorithms to solve large-scale OP instances, a heuristic and an exact algorithm, and experimentally show that both algorithms achieve outstanding results, both in terms of the quality of solutions and in terms of speed.

Esker Onak - Acknowledgments

Quiero comenzar agradeciendo a mis directores, María y Josean, el enorme apoyo académico y personal que me han proporcionado durante todos estos años. Gracias por vuestra constancia y dedicación, por los sabios consejos, por la confianza que habéis depositado en mí y especialmente, por sacar el akulu cuando hacía falta.

I would like to express my sincere gratitude to Prof. Grazia Speranza from the Operational Research Group at University of Brescia. It was truly an honor to be given the opportunity to work among such talented researchers during my three-month stay. I wish to thank all the members of the OR group for their warm hospitality and the excellent atmosphere. Thank you for all the lunch time talks, for recommending the places to visit, the gelato moments and for the inspiring seminars. Special mention goes to Andrea Mor who helped me enormously during my stay. Thank you for being so nice and attentive with me. You deserve the best!

I would also like to thank Dr. A. Duarte, Dr. J.J. Salazar-González and Dr. J. Silberholz, who provided me with their codes. I gratefully acknowledge the authors of the TSP solver Concorde for making their amazing code available to the public, since it has been the working basis of some of the algorithm implementations in this thesis.

Muchas gracias a Laureano Escudero, J.J. Salazar-González y a Dae-Jin por respaldarme con cartas de recomendación para la obtención de la la financiación que ha hecho posible esta tesis.

A los miembros del Grupo de Optimización Estocástica de la UPV/EHU, por aceptarme en el grupo aunque trabaje en problemas de optimización estocástica con varianza cero. Mención especial para Gloria, IP de varios proyectos de investigación de los que he recibido fondos para asistir a congresos internacionales. Siempre he admirado tu pasión por la programación y espero que sigas disfrutando de ella en el futuro. Quisiera agradecer a Araceli su disposición inicial para codirigir esta tesis. Gracias por tu compromiso con nuestra universidad. Agradecer también a los miembros de Inteligent Systems Group, en especial a Borja Calvo, por los ciclos de charlas en los que he participado y por todo lo aprendido de vosotros. Eskerrik asko Matematika eta Estatistika doktoretza programako arduradunei emandako aukeragatik, batez ere urte hauetan zuzendari izan diren Gustavo eta Javiri.

Hona iristeko nolabaiteko eragina izan duten pertsonak ere eskertu nahiko nituzke.

Fernando Garatea, matematiketarako grina sortu eta karrera egitera animatzeagatik. Carlos Gorria, masterrean enpresa praktikak egiteko aukera eskaini eta tesi honetan landutako problema ezagutzea ahalbidetzeagatik. Inma Arostegui, BCAMen sartzeko bidea zabaltzeagatik.

Me siento muy afortunado de haber compartido esta experiencia con gente tan valiosa. Gracias a los del grupo de Machine Learning. To my office mates who survived next to me in $9m^2$. Al staff de BCAM. A los que hemos compartido un café, una birra, un paseo... A los que comían a deshora. A los que habéis organizado cenas y salidas. A los apasionados de la programación. A los que sois pura actitud.

Bereziki eskerrak eman nahi dizkizuet lagun guztiei. Hor egoteagatik. Besarkada handi bana.

Tesi hau nire familiari eskaini nahi diot. Lan hau nekez gauzatuko zan zuongatik izango ez balitz. Nire guraso Lourdes eta Jose Antoniori, amuma Bene ta amuma Ramonari, anaia Joneri, Karmele eta Javiri: zuei, dana eta gehiago emategatik. Hau zortea nirea! Nire eredu zarien aitxitxe Juan eta aitxitxe Bixenteri. Nire gogoan zaudeten izeko Pili eta izeko Txarori. Nire famili guztiari.

Eta zelan ez, eskerrak bihotzez nire maitiek diren Esti eta Gariri. Bizi behar doguzenak!

Contents

CHAPTER 1

Introduction

Nothing is more challenging than a problem which is easy to understand but difficult to solve. Combinatorial Optimization, which optimizes discrete problems that emerge in a variety of fields, is in itself a field full of challenges. Such is the difficulty of these problems that Karp [1972] showed that many combinatorial problems are computationally intractable within the current computational paradigm. However, the need to solve relevant real-world problems has attracted many researchers to develop efficient algorithms.

The Travelling Salesperson Problem and the Knapsack Problem are two well-known combinatorial optimization problems. They are both easy to define, but difficult to solve. The research carried out for these two problems is a source of inspiration to solve other combinatorial problems. In this dissertation, we study the Orienteering Problem, a problem that can be seen as a combination of these two classical problems. Particularly, our objective is to develop algorithms to solve large Orienteering Problems.

1.1 The Orienteering Problem

The Orienteering Problem (OP), also called the Selective Travelling Salesperson Problem or the Maximum Collection Problem, is a routing problem proposed in the 80s, see Tsiligirides [1984] and Golden et al. [1987]. The name of the problem originates from a sports game, where the participants are given a topographical map with detailed checkpoints, each with an associated score, and a time limit. The participants who visit the checkpoints that maximize the total obtained score within the time limit, are the winners of the game.

THE PROBLEM
Given a weighted complete graph with vertex profits and a constant d_0, the goal is to find the simple cycle which, with a length not greater than d_0, maximizes the sum of the profits of the visited vertices.

Traditionally, the solution of the OP must visit a given edge or vertex of the graph. In the early works for OP, the solutions of the problem were paths starting and finishing in two given vertices. Finding a path whose ends are fixed is equivalent to finding a cycle which transverses the edge associated to starting and finishing vertices. In recent publications, it has become common to fix a vertex instead of an edge. Throughout all the dissertation, a feasible cycle solution for the OP must visit a given vertex, called the depot vertex. The OP can be modelled as follows:

$$\max \quad \text{total score of the vertices visited by } \tau \tag{1.1a}$$

$$\text{s.t.} \quad \tau \text{ is a simple cycle,} \tag{1.1b}$$

$$\tau \text{ has a length lower than } d_0, \tag{1.1c}$$

$$\tau \text{ visits the depot vertex} \tag{1.1d}$$

The OP can be seen as a combination of the Knapsack Problem (KP) and the Travelling Salesperson Problem (TSP). Given a set of items with an assigned weight and profit and a constant w_0, the goal in KP is to find the subset of items which, with a total weight lower than or equal to w_0, maximizes the sum of the profits of subset items. In the KP, the feasibility of a subset is checked in linear time. In the OP, however, the feasibility of a solution is checked by solving a TSP-decision problem. A subset of vertices is feasible if there exists a cycle (Hamiltonian in the subgraph obtained by the vertices) whose length does not exceed d_0, finding such a cycle is an NP-complete problem. This simple but non-trivial combination of two NP-hard problems makes the OP an interesting problem to study.

In Figure 1.1 we show the TSP solution (left) and the OP solution (right) for the instance pr76 of TSPLIB published in Reinelt [1991]. For the OP, the depot node is represented in green and the distance limitation is half of the TSP solution value on the left. The scores of the nodes are randomly generated as explained in Table 1.1.

The OP is classified as one of the three generic problems in TSPs with profits, see Feillet et al. [2005]. The TSPs with profits have two opposite criteria: one that motivates the salesperson to travel and another that imposes a constraint in the route length, e.g., the route must have a minimum length or the route length must be not greater than a given value. The other two problems of TSPs with profits are the Profitable Tour Problem (PTP) (Dell'Amico et al. [1995]) and the Price Collecting TSP (PCTSP) (Balas [1989]). In the PTP the goal is to maximize the difference between the total collected profit and the cost of the tour. Particularly, the PCTSP is closely related to the OP. In both problems, the solutions are simple cycles that contain a given depot vertex. The two problems differ in two aspects. First, the Knapsack constraint of the problem in the PCTSP is defined among the collected vertex profits rather than in the length of the route as in the OP. Secondly, the objective function in PCTSP is to minimize the route length while in the OP it is to maximize the collected vertex profits. See Angelelli et al. [2014b] for the study on the complexity and approximation algorithms for TSPs with profits.

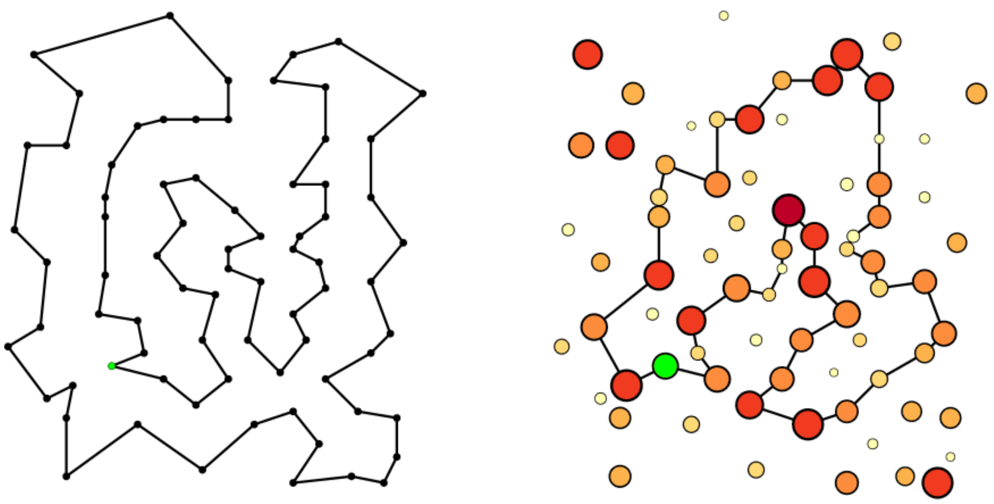


Figure 1.1: On the left, the TSP solution of pr76. On the right, the OP solution for pr76-Gen2-50

1.1.1 Complexity

NP-hard problem
The Orienteering Problem is an NP-hard problem since the existence of a polynomially bounded algorithm for it implies the existence of a polynomially bounded algorithm for well-known NP-complete problems, and hence for all NP-complete problems (Golden et al. [1987]).

In order to see that a polynomially bounded algorithm for the OP implies the existence of a polynomially bounded algorithm for NP-complete problems, it can be seen that a NP-complete problem is equivalent to some particular cases of the OP.

Given an undirected graph G, let us assign weight one to every edge and score one to every vertex, and let $|V|$ be the cycle length constraint of the OP. Then if the OP value is equal to $|V|$, there exists a Hamiltonian tour for the graph G. The Hamiltonian cycle problem, a known NP-complete problem which consists of determining the existence of a cycle that visits all the vertices in a graph exactly once, has a polynomially bounded algorithm if there exists such an algorithm for the OP.

It can also be seen that there exists a polynomially bounded algorithm for the TSP-decision problem if there exists such an algorithm for the OP. Given an undirected weighted graph G and a constant d_0, let us assign score one to every vertex. Then, if the solution value of OP is equal to $|V|$, there exists a tour for the graph G with length equal to or lower than d_0.

Brute force search algorithm

In a worst case scenario, using a brute force approach for the OP, we will need to evaluate all the simple cycles of the graph. If this is the case, for each non-empty subset of vertices, all the permutations of vertices in the subset need to be evaluated. Although the number of simple cycles might seem to be much larger than the number of Hamiltonian cycles, they are both comparable. Let n be the number of vertices of a graph and k the number of vertices in a simple cycle. Then, the number of simple cycles is:

$$\sum_{k=1}^{n} \binom{n}{k} k! = n! \sum_{k=1}^{n} \frac{1}{(n-k)!} = n! \sum_{k=1}^{n} \frac{1}{k!} \leq n! \sum_{k=0}^{\infty} \frac{1}{k!} = n!e \tag{1.2}$$

Hence, the brute force search algorithm for the OP has the same time complexity, $O(n!)$, as the brute force search algorithm for the TSP. However, its complexity is much bigger than the time complexity of the brute force algorithm for KP, which is $O(2^n)$.

1.2 Variants of the OP

Many practical problems have been modeled where the OP plays a crucial role. Some examples are travelling salesperson without enough time to visit all the cities (Tsiligirides [1984]), the home fuel delivery problem (Golden et al. [1987]), the tourist trip design problem (Vansteenwegen and Van Oudheusden [2007]; Souffriau et al. [2008]; Wang et al. [2008]), and the mobile-crowdsourcing problem (Yuen et al. [2011]).

In order to address these real-world problems, many variants of the OP have been proposed in the literature:

- Team OP (TOP): the goal is to determine M paths, each limited by a maximum length constraint, in order to maximize the total score. See Chao et al. [1996b], Boussier et al. [2007], Poggi et al. [2010], Dang et al. [2013], Keshtkaran et al. [2015], Bianchessi et al. [2018].
- OP with Time Windows (OPTW): each node has an assigned time window which determines when a node can be visited. See Vansteenwegen et al. [2009], Labadie et al. [2011], Gunawan et al. [2017].
- Arc OP (AOP): the profits are located in the arcs. See Archetti et al. [2016], Archetti et al. [2014a], Riera-Ledesma and Salazar-González [2017].
- Time Dependent OP (TDOP): the travel time between two nodes depends on the departure time. See Verbeeck et al. [2014].
- OP with Stochastic Profits (OPSP): the profits associated with the nodes are stochastic with a known distribution. See Ilhan et al. [2008].
- OP with Stochastic Travel and Service Times (OPSTS): the travel and service

times are stochastic. See Campbell et al. [2011].

- Generalized OP (GOP): each node has an assigned set of scores with respect to a set of attributes. See Geem et al. [2005] and Wang et al. [2008].
- Probabilistic OP: each node is available to visit with a certain probability. See Angelelli et al. [2017].
- Multi-agent OP: individual agents are self-interested in maximizing their score. However, the nodes have a capacity and can only receive a limited number of agents at the same time. See Chen et al. [2014].
- Clustered OP (COP): the nodes are clustered in groups. The score associated with each group is obtained when all the nodes in a particular cluster are visited. See Angelelli et al. [2014a].

In recent years, there has been a considerable increase in the publications related to OP. In Figure 1.2 we show the trend of the number of publications in which the OP is studied or used, according to Scopus.

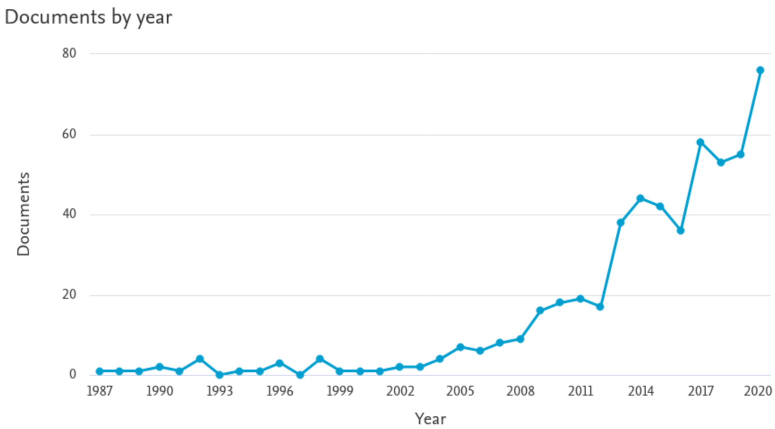


Figure 1.2: Trend of Orienteering Problem related publications. Source Scopus.

1.3 Benchmark instances for OP

Several benchmark instances have been proposed in the OP literature, which go from a few dozen of nodes in the early years to several thousands of nodes in more recent publications.

In the benchmark instances of the early works, it was common to use different starting and finishing nodes. In Tsiligirides [1984], the first paper dealing with the OP, three instances were presented, each with 21, 31 and 32 nodes. In Ramesh et al. [1992] 9 sets of nodes, with 10, 20, 30, 40, 50, 60, 80, 100, and 150 nodes, were generated using random scores and arc costs. In Chao et al. [1996a], two new sets of nodes were proposed, a square-shaped one with 66 nodes and a diamond-shaped one with 64 nodes.

In the last few decades, the OP approaches have been tested in instances where the starting and finishing nodes for the solution cycles are the same. Although the first set of instances of this kind is used in Laporte and Martello [1990], which consists of randomly generated instances involving from 10 to 90 nodes (the instances have not been published), the most influential benchmark instances for the OP are those proposed in Fischetti et al. [1998]. These instances are based on the well-known TSPLIB repository of benchmark instances for the TSP, see Reinelt [1991]. In this paper, the authors describe three methods of generating scores for OP instances from TSPLIB. In generation 1 (Gen1) all the vertices have score one. In generation 2 (Gen2), the scores are generated pseudorandomly as described in Table 1.1. In generation 3 (Gen3) the scores are proportional to the distance to the depot vertex. For all of these three generations the distance limitation is set as half of the TSP solution, $d_0 = \lceil 0.5 \cdot v(TSP) \rceil$. In this thesis, we have extended the set of benchmark instances to larger size problems. So far, instances up to 400 nodes of the TSPLIB had been evaluated in the OP literature; we also considered the ones involving up to 7397 nodes. The benchmark instances set is summarized in Table 1.1.

Table 1.1: Generations for instances based on TSPLIB.

Generation	Score for the ith node, $i \in [n]$	α	# medium $n \leq 400$	# large $n > 400$
Gen1	1	0.5	45	41
Gen2	$1 + (7141 \cdot (i-1) + 73) \mod 100$	0.5	45	41
Gen3	$1 + \lfloor 99 \cdot d_{1,i} / \max_{j \in [n]} d_{1,j} \rfloor$	0.5	45	41

All the instances used for the computational experiments are available in https://www.github.com/bcamath-ds/OPLib.

1.4 Review of the literature approaches for the OP

Since the publication of Golden et al. [1987], dozens of heuristic and exact approaches have been proposed to solve the OP. A review of the early approaches for the OP, prior to 1996, can be found in Chao et al. [1996a]. In the last decade, with the upsurge of variants and applications of the OP, new surveys have been published about the approaches, variants and applications of the OP. These recent surveys are Vansteenwegen et al. [2011], Gunawan et al. [2016] and Vansteenwegen and Gunawan [2019].

With the aim of setting a good starting point to introduce the contribution in this dissertation, we provide a background of the approaches proposed for the OP in the literature and describe the most important heuristic and exact approaches proposed thus far. In Table 1.2 we summarize the most influential heuristic approaches for the classical OP, while in Table 1.3 we summarize the exact algorithms developed for the OP. We have excluded from the lists the publications that do not compare the proposed algorithm with the benchmark instances in the literature or solve a different version to the classical problems, e.g., use an alternative objective function.

We name the benchmark instances using numbers to simplify the tables: [1] Tsiligirides [1984], [2] Laporte and Martello [1990], [3] Ramesh et al. [1992], [4] Chao et al. [1996a], [5] Fischetti et al. [1998] and [6] Gendreau et al. [1998b].

Regarding the heuristic approaches, we focus the summary on the used approach framework, solution initialization technique (Initialization), cycle length improvement heuristic (Length) and local search procedures (Local Search). We classify the initialization approaches into two groups: the constructive ones and the selective ones. By constructive, we mean techniques that add nodes, and paths, to the solution step by step. By selective, we refer to techniques that first determine the nodes in the initial solution and thereafter construct the cycle, or path, that transverses the selected nodes.

In Table 1.2, in order to simplify the alternative local search procedures, we follow the notation used in Keller [1989]. By (a, d), we mean that a nodes have been added and d nodes have been dropped from the solutions simultaneously. The most widely used local search is $(1, 0)$ where the added node is inserted with the cheapest insertion criteria. Since all these local search heuristics are inappropriate for the aim of solving large problems, we do not delve deeper in the specific proposed local search procedures. The primary reason why these local search procedures are not useful for large OP problems is that they are computationally expensive.

In the *Length* column we specify the heuristic used to optimize the cycle length of the solutions. These heuristics are: k-opt (Lin [1965]; Lin and Kernighan [1973]) and GENIUS (Gendreau et al. [1992]).

Note that all the proposed heuristics in the literature thus far share the property of working with feasible solutions. In these algorithms, whenever an unfeasible solution is obtained, the solution is amended to convert it to a feasible one.

Regarding the exact approaches for the OP, the most competitive approach thus far was proposed by Fischetti et al. [1998] two decades ago. To our knowledge, no exact algorithm for the classical OP has been published after this work. The first exact algorithm, a Branch-and-Bound (B&B) algorithm, was published in Laporte and Martello [1990] where bounds for the problem were provided based on the Knapsack relaxation of the OP. In Ramesh et al. [1992], new bounds for the B&B were obtained by Lagrangian relaxation. In Leifer and Rosenwein [1994] a Branch-and-Cut (B&C) algorithm was proposed, which included logical, connectivity, and cover cuts for the first time. In Gendreau et al. [1998b] a B&C was proposed for a variant of the OP which considers multiple depot nodes. The B&C approach in Fischetti et al. [1998] outperformed the previous ones in middle-sized OP instances by considering column generation, new valid inequalities (cycle cover and path inequalities), problem-specific separation algorithms, and an efficient primal heuristic.

1.5 Objectives of the thesis

The algorithms published so far in the literature were developed with small and medium-sized instances in mind. The main objective of the thesis is to design algorithms to solve large-sized OPs. With that aim, we plan to develop:

(i) A heuristic algorithm that, with low computational time, obtains solutions with acceptable quality:
   i. Minimize the need to check and recover the feasibility of solutions.
   ii. Initialize the solutions considering the relation between the distance constraint and the TSP solution value.
   iii. Improve the solutions with a local search procedure that scales for large-sized problem.

(ii) General techniques for Branch-and-Cut algorithms to solve large cycle problems:
   i. Safe shrinking of support graphs.
   ii. Techniques to speed up exact subcycle elimination separation algorithms.

(iii) A Branch-and-Cut algorithm able to obtain optimality certification in a wider set of instances than previous methods and to improve the known lower and upper bounds in the literature.

As a by-product of the previous goals we also pursue to:

(iv) Implement the algorithms and make the software publicly available.

(v) Create a repository with TSPLIB-based large-sized OP instances.

The rest of the thesis is organized as follows. In Chapter 2 an evolutionary algorithm is developed. We extend from the TSP the Edge Recombination crossover and a k-d tree-based local search is proposed. Chapter 3 introduces the cycle polytope and the shrinking technique. Three safe shrinking rules for the cycle polytope and two subcycle-safe shrinking rules are obtained. We extend efficient exact algorithms and procedures for the subcycle separation problem. In Chapter 4, a Branch-and-Cut algorithm is developed. In each of these three chapters, a section with computational experiment is included. In Chapter 5, we detail the structure, the installation and the use of the implemented software. In the appendices, pseudocodes and detailed experimental results are presented.

Table 1.2: Summary of heuristic approaches for the OP.

Publication	Approach	Initialization	Length	Local Search	Benchmark
Tsiligirides [1984]	Det. and Stoch.	constructive	2-opt	(1,0)	[1]
Golden et al. [1987]	Deterministic	constructive	2-opt	(1,0), center of gravity	[1]
Golden et al. [1988]	Stochastic	constructive	2-opt	(1,0), center of gravity	[1]
Keller [1989]	Specific	constructive	2-opt	(1,0), (1,1) and (1,2)	[1]
Ramesh and Brown [1991]	Specific	constructive	2-opt, 3-opt	(1,0) and (1,1) improvements	[1]
Wang et al. [1995]	Neural network	selective	2-opt	(1,0) and (0,1)	[1]
Chao et al. [1996a]	Heuristic	constructive	2-opt	(1,0), (1,1), (0, d)	[1], [4]
Gendreau et al. [1998a]	Tabu Search	constructive	GENIUS	(1,0), (a,d),tabu status	[6]
Tasgetiren [2001]	Genetic	selective	2-opt	injection crossover, penalized fitness	[1], [4]
				(1,0) (0,1) (1,1)	
Schilde et al. [2009]	ACO, VNS	constructive	2-opt	(1,0), (0,1), (1,1), (a,d), PR	[1], [4]
Silberholz and Golden [2010]	iterative	constructive	2-opt	(a,4)	[1], [4],[5]
Campos et al. [2014]	GRASP with PR	constructive	2-opt	(1,0), (1,1), PR	[4],[5]
Marinakis et al. [2015]	Memetic-GRASP	constructive	2-opt	(1,0), 1-point crossover	[1], [4]

Table 1.3: Exact Approaches for the OP.

Publication	Approach	Contributions	Benchmark
Laporte and Martello [1990]	B&B	KP bounds	[2]
Ramesh et al. [1992]	B&B	Lagrangian relaxation	[3]
Leifer and Rosenwein [1994]	B&C	Logical, Connectivity, Edge Cover	[1]
Fischetti et al. [1998]	B&C	Cycle Cover Path Inequalities Column Generation Primal Heuristics	[5]
Gendreau et al. [1998b]	B&C	Vertex Cover Alternative Obj Primal Heuristics	[6]

CHAPTER 2

EA4OP: An Evolutionary Algorithm for the OP

OUTLINE

In this chapter, we present an Evolutionary Algorithm for the OP. The key characteristic of the algorithm is to maintain unfeasible solutions during the search. Furthermore, it includes a novel heuristic for node inclusion in the route, an adaptation of the Edge Recombination crossover developed for the Travelling Salesperson Problem, specific operators to recover the feasibility of solutions when required, and the use of the Lin-Kernighan heuristic to improve the route lengths.

2.1 Introduction

There are several Evolutionary Algorithms (EA) proposed in the literature to solve OP and TOP Tasgetiren [2001]; Bouly et al. [2010]; Ferreira et al. [2014]; Marinakis et al. [2015]; Ostrowski et al. [2017], among others. In all of these approaches the solutions are initialized with constructive methods which add a new node to the route while the distance limitation constraint is satisfied and codified based on the visiting sequence of nodes. The tour lengths are improved using the 2-opt heuristic and general purpose genetic operators are adapted for the evolutionary part. Particularly, all of them use an adaptation of the single-point crossover or its generalization, the n-point crossover. Approaches Bouly et al. [2010]; Ferreira et al. [2014]; Marinakis et al. [2015] and Tasgetiren [2001], have been tested in the benchmark instances proposed by Chao et al. [1996a] (40 instances involving up to 66 nodes) and Tasgetiren [2001] (49 instances involving up to 33 nodes). Approach Ostrowski et al. [2017] has been tested in 90 TSPLib-based instances and 15 VRP-based instances proposed by Fischetti et al. [1998].

We propose a population-based evolutionary optimisation technique whose main characteristic is to maintain unfeasible solutions during the search process. Essentially the algorithm follows the steady-state genetic algorithm schema Whitley et al. [1989] with the difference that, at some generations, we perform a tour-improving procedure followed by node dropping and adding strategies, for feasibility conversion and path tightening respectively. The pseudocode is described in Algorithm 1.

Our approach, in addition to the common parameters of any genetic algorithm (population size, mutation probability), uses a specific parameter, *d2d*, that controls the frequency of the feasibility and improving phase.

Algorithm 1: Evolutionary Algorithm

```
Build initial population (2.3.1);
Tour improvement (2.3.3);
Drop operator (2.3.4);
Add operator (2.3.5);
i=0;
while stopping criteria are not satisfied OR mod(i, d2d) ≠ 0 ) do
    i=i+1;
    if mod(i, d2d) ≠ 0 then
        Select two parents (2.3.2);
        Crossover (5);
        Mutation (14);
        if child better than worst in the population then
            Insert the child in the place of the worst individual;
        end
    else
        Tour improvement (2.3.3);
        Drop operator (2.3.4);
        Add operator (2.3.5);
    end
end
```

2.2 Solution Codification

A solution to the problem can be seen as a sequence defined by a subset of nodes (route).

Codification

In order to codify that solution, a permutation of the whole set of nodes has been considered, $\pi = (\pi_1, \dots, \pi_n)$. In this permutation, π_i represents the next node visited after vertex i in the route. The nodes in the route form a cycle in the permutation and a node which is not in the route is codified as a fixed point, i.e, $\pi(i) = i$.

Figure 2.1 shows a solution of an OP whose associated codification is the following permutation, $\pi = (6, 2, 3, 4, 8, 7, 5, 1)$. Note that the nodes in the solution route

$\{1, 6, 7, 5, 8, 1\}$ form a cycle in the permutation π, while those nodes off the route $\{2, 3, 4\}$ are fixed points in the permutation.

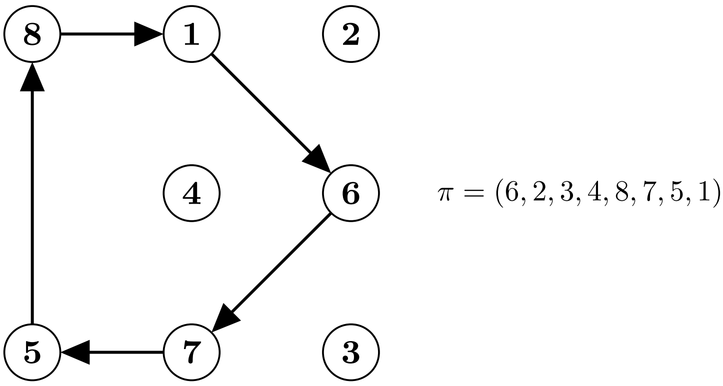


Figure 2.1: Example of a solution in an eight-node graph and its corresponding codification as a permutation.

Note that not every possible permutation is a valid solution of the problem: first, the route length limitation constraint may not be satisfied; secondly, sub-routes may also appear. However, this route codification has been chosen for implementation reasons. On one hand, a fixed length codification was desirable; on the other hand, some operations over the solutions, such as checking if a node is contained in the route, can be efficiently implemented using this codification. A similar codification was previously proposed for the Prize Collecting TSP Balas [1989].

2.3 Components

2.3.1 Initial population

Algorithm 2 shows a pseudocode for generating *npop* individuals for the initial population. An individual is generated in two steps. In the first step, a subset of nodes is randomly chosen, where each node is sampled with probability p. In the second step, a route passing through the subset of nodes is randomly created and codified as described in Section 2.2.

Algorithm 2: Initial population

for $i = 1$ *to* $npop$ **do**
    v_1 node is included in the subset of nodes;
    for $j = 2$ *to* n **do**
        v_j node is included in the subset of nodes with probability p;
    end
    Construct a tour by randomly ordering the selected nodes;
end

The probability p is a parameter of the algorithm, where $n \cdot p$ determines the expected number of visited nodes of each generated individual. In addition, note that the obtained initial individuals could be unfeasible.

2.3.2 Genetic Operators

In this section we will describe the genetic operators - parent selection, crossover and mutation - that are used to evolve the population. While the chosen parent selection operator is a general purpose selection procedure, the crossover and mutation operators have been specifically developed or adapted for the OP problem.

Parents selection

Our selection operator is a kind of hybridisation between tournament and roulette wheel selection, see Algorithm 3. In the first step, *ncand* candidates are uniformly at random selected from the population. In the second step, the roulette wheel selection is carried out, based on the individual fitness (i.e., its objective function or total score), where a correction is performed (subtraction of the minimum fitness) in order to point out the fitness quality differences between candidates.

Algorithm 3: Parents selection

Select uniformly at random *ncand* candidates from the population, $C = \{I_1, \ldots, I_{ncand}\} \subset \{1, \ldots, npop\}$;
Compute $m := min_{I_i \in C}(f_{I_i})$, where f_{I_i} is the objective function value of $I_i \in C$;
Compute $r_i := f_{I_i} - m + 1$, $i = 1, \ldots, ncand$;
Compute $p_i := \frac{r_i}{\sum r_i}$, $I_i \in C$;
Sample twice with the distribution $(p_1, \ldots, p_{ncand})$ to obtain two parents;

Crossover operator

The crossover produces a new child solution from a given pair of parents solutions by using an adaptation of the well-known Edge Recombination operator Whitley et al. [1989].

CROSSOVER GOALS
In the OP, we are interested in inheriting two main characteristics from the parents related with the nodes and the edges. Regarding the visited nodes, we want to maintain all the nodes that are common to both parent solutions, to include, with a probability, the nodes that belong to only one parent, and to exclude the nodes that do not belong to any parent solution. Regarding the route length, we want to use as many edges of the parents as possible in order to pass on the maximum amount of information and decrease length quality losses in the new child solution.

The original ER crossover Whitley et al. [1989] was designed for problems where the solution space consists of Hamiltonian cycles; now we have extended it for a larger set of sequencing/ordering problems, where the solution space consists of simple cycles which do not necessarily contain all the nodes. This generalisation does not use the information of the associated cost of the edges and, therefore, it is possible to produce unfeasible solutions for the OP.

The operator uses the so-called edge map, which is a summary of parental information, to guide the procedure. The **edge map** contains, for each common node of the parental graph, its degree, connected nodes and intermediate paths. Representing the route of the first and second parent as the graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ respectively, the **parental graph** consists of all vertices and edges arising in the solutions of the two parents, i.e., $PG = (V_1 \cup V_2, E_1 \cup E_2)$. A node u is a **common node** of the parental graph if that vertex belongs simultaneously to both parents, $u \in V_1 \cap V_2$. A node u is a **connected node** of a node v if u and v are common nodes and there exists a path over the parental graph connecting both nodes which does not contain a third common node. The **degree** of a common node is the number of nodes which are connected to it. An **intermediate path** between two common nodes u and v is any path from u to v, which is inside the parental graph with no more common nodes.

The ER crossover operator builds the child route as follows (see Algorithm 4): first, the edge map is constructed, and the starting node v_1 is assigned to be the **current node**. Each time the current node is reassigned, it is removed from the edge map, and the degree of each non-visited common node is recomputed. At each step we will decide which the next common node to visit is by selecting from the set of the non-visited connected nodes of the current node the one that has the lowest degree, where ties are broken randomly. If we reach a node whose all connected common nodes are already visited, we will choose the next node randomly from the set of non-visited common

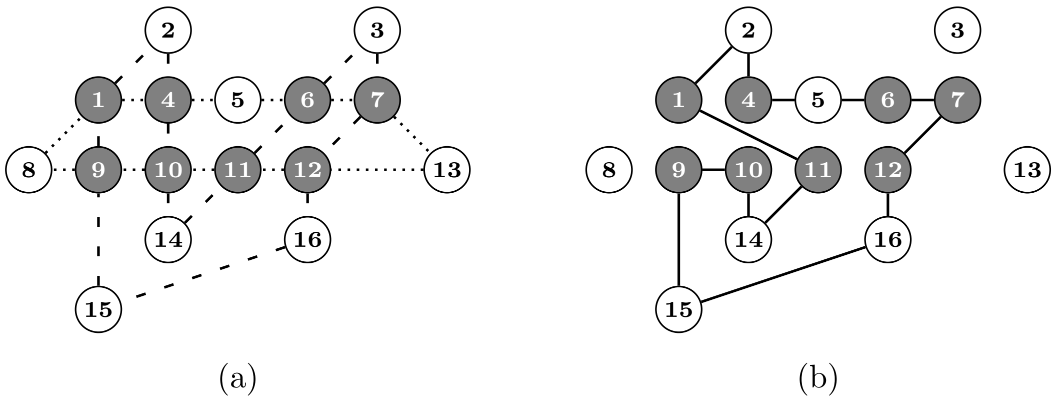


Figure 2.2: Example of a crossover. (a) Parental graph. The route of the first parent is represented by dotted line, the route of the second parent with dashed line. The common nodes are filled in gray. (b) Child after the crossover.

nodes. A intermediate path between the current node and next node is randomly chosen and its nodes incorporated to the route. The process finishes when there are no more common nodes left to visit.

Note that the operator does not make sense when there is a unique common node, v_1, or when the solution routes are equal. In any of these cases, the crossover procedure is skipped, and one of the parent solutions is cloned.

In Figure 2.2, two parents solutions are shown (a) and the child (b) produced after the ER crossover application. Table 2.1 shows the associated edge-map and Figure 2.3 shows some illustrative steps of the operator. The algorithm starts at common starting node 1. Both of its connected nodes, 4 and 9, have degree two - we have already removed the node 1 from the edge map-, so the algorithm will make a random choice. Assume that the common node 4 is chosen, and again randomly we choose one of the possible paths to reach the node 4, in this case we assume that the path chosen is $(1, 2, 4)$, see first step in Figure 2.3. The candidates for the next common node are 6 and 10. Both have degree 2 so randomly choose one, assume that 6 is chosen. There is a unique path to choose that goes from 4 to 6, the one that passes through node 5, see second step. In the last step, all the common nodes have been visited so the algorithm will join node 11 and 1.

Table 2.1: Example of an edge map

Common Node	Connected Nodes	Degree	Intermediate paths
1	4	2	(1,4), (1,2,4)
	9		(1,9), (1,8,9)
4	1	3	(4,1), (4,2,1)
	6		(4,5,6)
	10		(4,10)
6	4	3	(6,5,4)
	7		(6,7), (6,3,7)
	11		(6,11)
7	6	2	(7,6), (7,3,6)
	12		(7,12), (7,13,12)
9	1	3	(9,1), (9,8,1)
	10		(9,10)
	12		(9,15,16,12)
10	4	3	(10,4)
	9		(10,9)
	11		(10,11), (10,14,11)
11	6	3	(11,6)
	10		(11,10), (11,14,10)
	12		(11,12)
12	7	3	(12,7), (12,13,7)
	9		(12,16,15,9)
	11		(12,11)

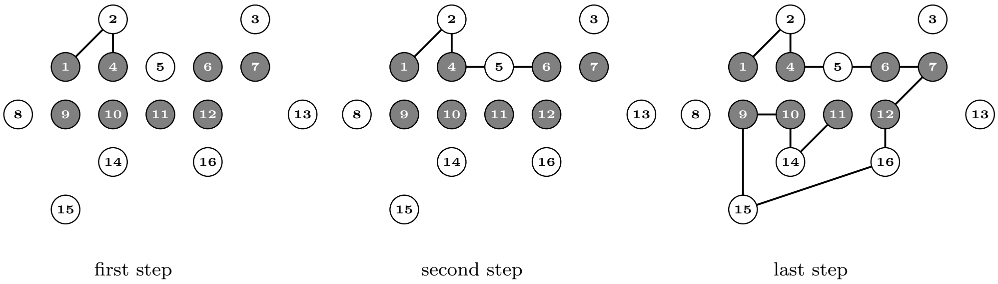


Figure 2.3: Illustration of the crossover operator. The left and center figures show the results of two consecutive steps of the crossover algorithm, while the right figure represents the last step before closing the route.

Algorithm 4: ER crossover operator

Initialize *current node* to v_1;
while *there are non-visited common nodes* **do**
  Remove all the occurrences of *current node* from the connected nodes of edge map;
  if *at least one connected node of the current node is not visited* **then**
    Update *next node* as the connected node with the smallest degree, ties are broken randomly;
    Choose randomly an intermediate path between *current node* and *next node*. Insert the path after the *current node*;
    Rename the *next node* as the *current node*;
  else
    if *there are non-visited common nodes* **then**
      Select randomly a non-visited common node and insert it on the route after the *current node*;
      Call it the *current node*;
    end
  end
end

Mutation operator

At each generation, after the crossover operator has been applied, a mutation is performed, see Algorithm 5. To perform the mutation, we will choose a node uniformly at random from $\{v_2, \ldots, v_n\}$. If the node is on the route, the node is dropped and its adjacent nodes are connected. If the node is not on the route, it is inserted in the best place - using the same heuristic explained later in the add operator 2.3.5.

Algorithm 5: Mutation operator

Select uniformly at random a node from $\{v_2, \ldots, v_n\}$;
if *the node is on the route* **then**
  Remove the node from the route and connect the adjacent nodes;
else
  Insert the node on the route, using the heuristic explained in Section 2.3.5;
end

2.3.3 Tour improvement operator

The feasibility of a solution closely depends on the order of the visiting nodes. A set of nodes could belong to a feasible or an unfeasible solution, depending just on the ordering of them on the route. The aim of this operator is to decrease the length of the routes as much as possible. In this manner, first, unfeasible solutions are attempted to convert to feasible solutions, second, the lengths of the feasible solutions are decreased in order to insert new nodes during the add operator 2.3.5.

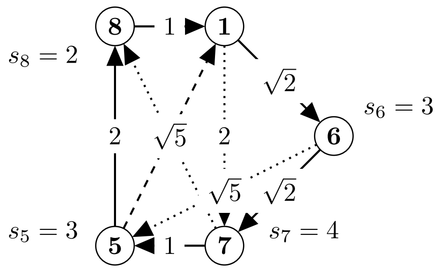
Finding the shortest route for a subset of nodes is equivalent to solving a TSP for that set of nodes. In the extensive literature that has the TSP, there is a vast quantity of heuristic approaches that can be used for the OP. We are particularly interested in those local search techniques that provide a high quality solution in a reasonable time due to the fact that the tour improvement is applied many times during the algorithm. In this work we have considered the Lin-Kernighan heuristic Lin and Kernighan [1973]; Applegate et al. [2007].

2.3.4 Drop operator

Improving the tour length might not be enough to convert an unfeasible solution to a feasible one, it could still continue violating the route length limitation constraint. In this case, in order to obtain a feasible solution, it is necessary to delete nodes from the solution until it fits the distance limitation.

To that end, we sort the nodes contained in the route considering both the cost in terms of length and the fitness gain for visiting each node. Namely, we want to drop the nodes that concurrently have a low contribution to the fitness and are costly to visit. Let us define the value for sorting the visited nodes as $drop(v_i) = \frac{s_i}{d_{i_p,i}+d_{i,i_n}-d_{i_p,i_n}}$ where v_{i_p} and v_{i_n} nodes are the previous and next nodes to v_i, respectively (see the drop operator in Algorithm 6 and the example in Figure 2.4).

Thereby, at each step of the drop operator the node with the lowest *drop* value is removed from the solution. The algorithm stops once it obtains a feasible solution.



$$
\begin{aligned}
drop(v_5) &= 3/(2+1-\sqrt{5}) &\sim 3.93 \\
drop(v_6) &= 3/(\sqrt{2}+\sqrt{2}-2) &\sim 3.62 \\
drop(v_7) &= 4/(\sqrt{2}+1-\sqrt{5}) &\sim 22.45 \\
drop(v_8) &= 2/(1+2-\sqrt{5}) &\sim 2.618
\end{aligned}
$$

Figure 2.4: Drop operator example. After evaluating the *drop* value of each node, the node 8 is removed from the tour.

Algorithm 6: Drop operator

```
1 while NOT distance limitation constraint is satisfied do
2     Order nodes according to drop index and remove the one with the lowest value;
      Update route length and fitness;
3 end
```

2.3.5 Add operator

Once the individual has been made feasible, we apply an improvement mechanism to it. It consists of the addition of new nodes to the current route. This operator is applied for node inclusion while the distance limitation constraint is satisfied, see Algorithm 7.

When dealing with node insertion, we have to set some criteria in order to select the most suitable node to add to the route and, then, to decide where the insertion should be made.

Before defining the insertion criteria, let us define an associated value, $addcost(v_i)$, for each non-visited node, v_i, that approximates the increase of the route length when inserting it to the route. A common heuristic that appears in the literature in order to calculate the *addcost* value is to evaluate the cost of each possible insertion in the route and to take the minimum value Campos et al. [2014]; Silberholz and Golden [2010].

CHEAPEST INSERTION APPROACH

If m represents the number of visited nodes in a solution, then the computational cost of selecting the candidate to insert at each step when using the cheapest insertion approach is $O((n-m) \cdot m) \leq O(n^2)$. Using the information calculated in the first step, i.e., the insertion position and the *addcost* of each non-visited candidate, it is possible to decrease the computational cost of selecting a candidate for the rest of the steps. This way we have $O((n-m) \cdot m)$ for the first step and $O(n-m)$ for the rest of the steps.

Although the previous method is quadratic, a faster node insertion method is still desirable, since a large amount of queries of this type are made during the algorithm. Therefore, we propose a new heuristic method for node insertion, one that speeds up the process at the expense of the quality of the *addcost* approximation.

To evaluate the inserting cost of a non-visited node, we start by finding the three nearest visited nodes. If two of these three nearest nodes are adjacent in the route, the *addcost* value is the cost of inserting the candidate node between these two nodes (see Figure 2.5). When there are more than two pairwise adjacent nodes in the 3-nearest set, the *addcost* value is determined by the choice that minimises the adding cost. Otherwise, if none of the three nearest nodes are adjacent to each other, calculate the cost of inserting the candidate node between the contiguous nodes of the three nearest nodes. There are six different options, so we choose the one with the minimum value for the *addcost*.

Because of the design of the proposed minimum cost insertion heuristic, when the distance matrix is given by spatial points, the computational cost can be decreased using a data structure to accelerate the proximity queries. In our case, we have used a k-d tree, which is built once in the whole algorithm.

Finally, the *addvalue* is defined to set the inserting preference of a non-visited node using the *addcost* and the score of the node. The inserting preference of a non-visited node depends whether the insertion is feasible or not. When the insertion is feasible, i.e., the current length plus the *addcost* value is not greater than the route length limit, the inserting preference is defined as $addvalue(v_i) = s_i/addcost(v_i)$. When the insertion is not feasible, the inserting preference of the node is set to 0. If the maximum value of *addvalue* is positive, the node which maximizes the *addvalue* is inserted in the route, and the process is repeated. The add operator stops when adding any of the non-visited nodes leads to an unfeasible solution, i.e., when all the *addvalue*s are 0.

k-d tree based 3-nearest insertion approach

When the nodes are spatial points, for the insertion approach, we use k-d trees for semidynamic point sets [Bentley, 1990]. The k-d tree is build in $O(n \log n)$ time. For each non-visited node, the k-d tree is updated (undelete and delete the node) in $O(1)$ time and the three nearest nodes are found in $O(\log(n))$. Therefore, the total cost of finding the best position in the route for all the non-visited nodes in the first step is $O((n-m)\log(n)) \leq O(n \log n)$.

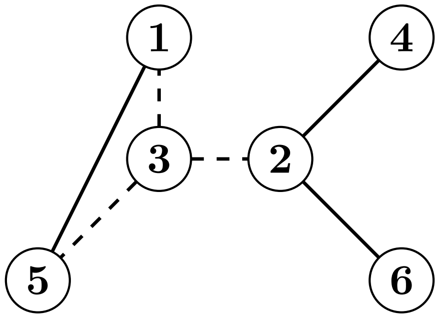


Figure 2.5: Example of an evaluation of the cost of inserting a node in the route. Node v_3 is the node to evaluate and the rest of nodes are part of the route (solid line). In this case, the best position for the node v_3 in the route is between the adjacent nodes v_1 and v_5.

In Figure 2.5 we represent the calculation of the heuristic to obtain the *addcost* of the non-visited node v_3. First, we search the three nearest nodes from v_3 on the route, in this case v_1, v_2 and v_5. Given that v_1 and v_5 are adjacent in the route, we assign to v_3 the increase of the distance route if v_3 is added between v_1 and v_5, i.e., $addcost(v_3) = d_{1,3} + d_{3,5} - d_{1,5}$.

2.3.6 Stopping criteria

There are two main stopping criteria for our evolutionary algorithm. The first one is based on the distribution of the population fitness. Specifically, the algorithm stops when a certain proportion of the solutions has the same fitness as the best solution of the population. The second one is a limitation on the execution time.

These criteria are evaluated after the feasibility of the solutions is checked and the add operator is performed, particularly, when the generation number is a multiple of *d2d*.

2.4 Computational results for EA4OP

This section presents the results of the computational experiments carried out for testing the evolutionary algorithm explained in the previous section. The proposed approach has been compared with the exact branch-and-cut algorithm (B&C) Fischetti et al. [1998] and three state-of-the-art heuristics: GRASP with Path Relinking (GRASP-PR) Campos et al. [2014], tabu search (TS) Gendreau et al. [1998a] and the two-parameter interactive algorithm (2-P IA) Silberholz and Golden [2010]. Results for TS are not reported because they were not competitive compared with the rest of the approaches, but they are available upon request from the authors. The benchmark instances have been generated from those obtained from TSPLib repository. We have split up the instances into two groups: medium-sized instances, up to 400 nodes and large-sized

Algorithm 7: Add operator

while *NOT stop* **do**
  for *node* v_i *not in route* **do**
    Get the three nearest nodes in the route for v_i, $V_3^i = \{v_1^i, v_2^i, v_3^i\}$;
    if *at least two nodes of* V_3 *are adjacent in the route* **then**
      Find the pair (v_{prev}, v_{next}) where $v_{prev}, v_{next} \in V_3$ that are adjacent in the route that minimizes $d_{prev,i} + d_{i,next} - d_{prev,next}$;
    else
      Define:;
      $V_3^* = \{(v_1^i, v_{1_{next}}^i), (v_2^i, v_{2_{next}}^i), (v_3^i, v_{3_{next}}^i)\} \cup \{(v_{1_{prev}}^i, v_1^i), (v_{2_{prev}}^i, v_2^i), (v_{3_{prev}}^i, v_3^i)\}$;
      Find the pair $(v_{prev}, v_{next}) \in V_3^*$ that minimizes $d_{prev,i} + d_{i,next} - d_{prev,next}$;
    end
    $addcost(v_i) = d_{prev,i} + d_{i,next} - d_{prev,next}$;
    if *route length* + $addcost(v_i) \leq d_0$ **then**
      $addvalue(v_i) = s_i / addcost(v_i)$;
    else
      $addvalue(v_i) = 0$;
    end
  end
  Select the node, v_i, which maximizes *addvalue*;
  if $addvalue(v_i) > 0$ **then**
    Include the selected node in the route;
    Update route length and fitness;
  else
    Stop;
  end
end

ones, up to 7397 nodes. As detailed in the literature, three generations are classified according to the definition of scores. A fourth generation has been created with the most difficult instances for the exact methodology.

The solution quality and the computational cost have been analysed. The solutions have been measured in terms of the quality gap (gap), defined as the relative difference in percentage between the best known or optimal solution (opt) and the solution of the corresponding algorithm ($best$), i.e., $gap = 100 \cdot \frac{opt-best}{opt}$. The computational cost in seconds is measured via time consumption.

The computational experiments are reported as follows. The validation of the proposed algorithm components is carried out in Section 2.4.2. Section 2.4.3 shows the performance of the evolutionary algorithm versus the exact B&C and two state-of-art heuristics: GRASP-PR and 2-P IA. The detailed numerical results are available in Appendix B.1.

2.4.1 Parameter and heuristic selection

In order to perform the parameter and heuristic selection, we have selected five medium-sized instances of generation 2, involving the largest amount of nodes without repeating the "family" (gil262, a280, lin318, pr299 and rd400). We have chosen instances from generation 2 precisely because, contrary to the other generations, here the scores are pseudo-randomly generated.

Solution initialization parameters

As explained in Section 2.3.1, an initial solution is generated in two steps: the first one consists of randomly selecting a subset of nodes to be visited; the second one consists of constructing the tour involving the selected nodes, i.e., giving an order in the selected subset of nodes. It is desirable that the average number of selected nodes in a solution, $n \cdot p$, is close to the number of nodes in the optimal solution. A straightforward choice is to select p as the proportion between the distance limit and the TSP solution, i.e., $p = d_0/v(TSP)$. However, the results achieved by the B&C in Fischetti et al. [1998] show that the optimal solution tends to visit a higher number of nodes than expected. Therefore, we have decided to overestimate the number of initial nodes using $p = \sqrt{d_0/v(TSP)}$. To approximate the TSP value, Lin-Kerninghan heuristic has been used and this computational time has been considered in the global time of the algorithm.

In order to get an idea of the influence of the parameter p on the population initialization and on EA4OP, we have tested three different choices of p: α^2, α and $\sqrt{\alpha}$ where $\alpha = d_0/v(TSP)$. The rest of the parameters have been set as detailed in Section 2.4.1. The experiments show that the best mean gap either for the initialization or for the EA4OP is obtained using $p = \sqrt{\alpha}$. Furthermore, in the initialization, the closest solutions to the optimum in terms of visited number of nodes are obtained using the parameter value mentioned. However, it is interesting to note that the higher the value of p used, the longer the time that is needed to initialize the population, due to the higher amount of nodes included in TSP problems that are solved during the initialization, see details in supplementary material.

Genetic operator parameters

The parameters value selection for the algorithm (*ncand*, *npop*, *d2d* and *pmut*) has been performed using non-parametric statistical tests: Friedman test for multiple (more than two) mean comparisons, Wilcoxon signed-rank test for two mean comparisons and Finner post-hoc test for pairwise comparisons García et al. [2010]. For all of these tests, 0.05 has been used as significance level.

Taking into account that, depending on the target (gap or time), the selected values for the parameters might differ, gap has been prioritized over time. Therefore, the analysis is performed in two steps: in the first step, a Friedman test on the gap is carried out for each parameter. If it does not find significant differences for the parameter values, all of the values are considered for the next step. Otherwise if it finds significant differences between the achieved gaps by the different values, we continue with Finner post-hoc tests to select the values that obtain the best gaps. Those values which are not significantly different from the best gap are considered for the second step. If all values have significant differences with the best, this is the value chosen and the procedure finishes here for that parameter. In the second step, previously selected values are considered and the procedure detailed for the gap is repeated now for the time. In the case that there are several parameter values with no significant differences with the value that obtains the best mean time, the value with the lowest mean gap is chosen.

For each parameter, the following set of values has been considered: $ncand \in \{5,\ 7,\ 10\}$, $npop \in \{10,\ 20,\ 50,\ 100\}$, $pmut \in \{0.01,\ 0.05,\ 0.1\}$ and $d2d\ \in \{5,\ 10,\ 20,\ 50\}$ where $d2d < npop$. For each parameter a univariate analysis has been conducted, except for *npop* and *d2d* - for which a bivariate analysis has been carried out. The values *(npop, d2d)*=(100, 5) and (100, 10) have been excluded from the analysis because those configurations require an excessive amount of time. Each combination of the parameters has been run 10 times.

Table 2.2: Statistical hypothesis testing for parameter selection

		Gap			Time		
Parameter	Value	Mean	Friedman p-value	Post Hoc corrected p-value	Mean	Friedman p-value*	Post Hoc corrected p-value
ncand	10	5.184	$< 2 \cdot 10^{-16}$	-	5.287		
	7	5.521		$6.5 \cdot 10^{-04}$	5.941		
	5	6.209		$< 2 \cdot 10^{-16}$	6.262		
(npop, d2d)	(100, 20)	2.559	$< 2 \cdot 10^{-16}$	-	6.910	$5.7 \cdot 10^{-14}$	
	(100, 50)	2.732		0.280	4.403		
	(50, 5)	4.281		$8 \cdot 10^{-07}$	5.146		
	(50, 10)	4.326		$4 \cdot 10^{-08}$	2.863		
	(50, 20)	4.390		$1 \cdot 10^{-09}$	1.949		
	(20, 5)	8.243		$< 2 \cdot 10^{-16}$	0.811		
	(20, 10)	8.577		$< 2 \cdot 10^{-16}$	0.485		
	(10, 5)	16.56		$< 2 \cdot 10^{-16}$	0.157		
pmut	0.01	5.583	0.57		5.692	$4.4 \cdot 10^{-03}$	-
	0.05	5.725			5.8702		0.583
	0.1	5.606			5.9279		$6 \cdot 10^{-03}$

*: Wilcoxon signed-rank test has been used to compare (100,20) and (100,50).

Table 2.2 details the mean gaps, the mean times and the p-values of the tests obtained during the selection procedure. For instance, based on this information, we have set *pmut* parameter to 0.01. In the first step, using Friedman test we obtained that there are no significant differences in terms of gap between the values of *pmut*, therefore, all the values were considered for the next step. Regarding the time, Friedman test gave that there exist significant differences between the *pmut* values, so we proceed with the Finner post-hoc test. Finally, parameter value 0.01 is selected since comparing the gap between *pmut* values with no significant differences in terms of computation times it obtains the lowest gap. After the statistical tests, the following parameter values were chosen for the computational experiments: $ncand = 10$, $npop = 100$, $d2d = 50$ and $pmut = 0.01$.

Tour improvement operator

Preliminary experiments in the 5 instances of the previous training set with 2-opt, 3-opt and Lin-Kernighan approaches showed that the Lin-Kernighan technique as TSP local search was the most suitable. We appreciated that the solutions obtained for the OP using this method were better than with the rest of the techniques, while the time needed to accomplish the search was not substantially larger.

Stopping criteria

As explained in Section 2.3.6, there are two stopping criteria. The first one, which is based on the distribution of the fitness, stops the algorithm when the first quartile of the population's fitness is the same as the best solution fitness. The second one, which is a time limitation, stops the algorithm when the execution time exceeds 5 hours.

2.4.2 EA4OP components validation

In this section, we verify that all the components in the EA4OP are necessary to obtain high quality solutions. We have implemented three algorithms in order to evaluate the contribution of the components in the EA4OP algorithm.

- Algorithm 3.3.1: This algorithm builds a large random population and applies the drop and add operators to each individual. We have considered the average number of solutions used by the EA4OP to set the size of the population, *npop*, for Algorithm 3.3.1, for each instance and generation. As we are using a steady-state algorithm, the amount of solutions used in a run of the EA4OP is equal to the initial population size plus the number of iterations. Algorithm 3.3.1 is used to evaluate the contribution of the evolution process of our algorithm.
- Algorithms 3.3.2 and Algorithm 3.3.3: In these algorithms, we consider a EA4OP but without the crossover. Instead of selecting two parents and crossing them,

we select only one parent using the procedure of the parents selection operator and applying the mutation operator. Two different versions of this algorithm have been tested, both of which differ in the relaxation of the stopping criteria. Algorithm 3.3.2 stops when all the solutions of the population have the same fitness. Since Algorithm 3.3.2 obtains lower computation times than EA4OP, we have also considered Algorithm 3.3.3, which is similar to Algorithm 3.3.2 but stops when the computation time reaches the mean time used by EA4OP. Algorithms 3.3.2 and 3.3.3 are used to evaluate the contribution of the ER crossover operator.

In order to perform the comparison of these algorithms, all of them have been configured with the same parameters used in EA4OP ($ncand = 10$, $npop = 100$, $d2d = 50$ and $pmut = 0.01$), except the parameter *npop* for Algorithm 3.3.1, which has been explained above. These algorithms have been run in five medium-sized and five large-sized instances of each generation, which have been selected using the same criteria as in Section 2.4.1.

Table 2.3: Comparison between the results of Algorithm 3.3.1, Algorithm 3.3.2, Algorithm 3.3.3 and EA4OP.

		Algorithm 3.3.1		Algorithm 3.3.2		Algorithm 3.3.3		EA4OP	
Generation	Size	Gap	Time	Gap	Time	Gap	Time	Gap	Time
Gen1	Medium	13.75	10.40	10.71	1.81	9.63	4.64	1.76	4.54
	Large	15.19	15468.30	10.07	1527.96	7.44	5501.40	0.00	5500.71
Gen2	Medium	13.20	11.96	9.01	1.75	8.43	5.05	1.21	4.93
	Large	16.33	16887.87	10.80	1721.30	5.93	6397.23	0.00	6397.06
Gen3	Medium	14.58	12.46	11.32	1.95	10.08	5.45	3.69	5.04
	Large	17.15	17635.65	10.94	1719.08	7.95	6241.82	0.00	6241.36
Gen4	Medium	2.16	6.89	1.28	1.79	1.24	5.66	0.07	5.14
	Large	16.75	17095.59	10.82	1490.95	7.23	3498.83	0.00	3498.29

The results are summarized in Table 2.3. They show that building a large random population needs a large amount of time while it does not obtain competitive results in terms of solution quality. This large amount of time is due to the requirements for making a random population feasible. It can be concluded that the proposed evolution speeds up the generation of individuals. Furthermore, it is essential to obtain high quality solutions.

Table 2.3 also shows that in most of the instances Algorithm 3.3.3 improves the gap results of Algorithm 3.3.2, however they are still not competitive with those obtained by EA4OP. Therefore, it can be assumed that the proposed adaptation of the ER crossover operator has an important contribution in the EA4OP.

In view of these results, we assume that the contribution of the evolutionary part and, specifically, the proposed adaptation of the ER crossover are essential in the overall algorithm.

2.4.3 Comparison with state-of-the-art algorithms

The experiments have been run on a workstation with Intel(R) Xeon(R) CPU E5-2609 v3 @ 1.90GHz processor using a single thread and a maximum of 4 GB RAM. For the experiments of this chapter, we have defined a new generation method (generation 4) involving instances with $d_0 \neq \lceil 0.5 \cdot v(TSP) \rceil$. With that in mind, we have considered the instances with scores of generation 2 and created all the cases with $d_0 = \lceil \alpha \cdot v(TSP) \rceil$, where $\alpha \in \{0.05,\ 0.10,\ \ldots,\ 0.45,\ 0.55,\ \ldots,\ 0.95\}$. From these 18 cases we have chosen the most difficult instance for the B&C in Fischetti et al. [1998]. When all the problems finish before the time limitation for the B&C, we choose the α whose solution takes the longest time. Otherwise, when at least one problem reaches the time limitation, we choose the α whose solution takes the longest separation time at the end of the time limitation for the B&C .

The evolutionary algorithm for OP (EA4OP) was implemented in C language. We have reused the code from the Concorde TSP solver for the routines related to dynamic k-d trees and the Lin-Kernighan TSP method. The source code has been published with a GPLv3 license, except the third-party code mentioned above, which has an academic license. The code is available at `https://github.com/bcamath-ds/compass`.

For comparison purposes, the following algorithms have been tested: the exact B&C algorithm from Fischetti et al. [1998] and two heuristics: GRASP-PR Campos et al. [2014] and 2-P IA Silberholz and Golden [2010]. For each heuristic, 10 runs have been performed at each instance, while the exact algorithm has been run once. All the experiments have been performed under the same conditions: the same machine, the same language (C) and the same compiler (gcc 4.8.5) with the same flags (-O3).

For a fair comparison in terms of computational time, the results of the B&C algorithm have been obtained with CPLEX 12.5.0 instead of the original LP solver CPLEX 3.0. Note that the papers Campos et al. [2014] and Silberholz and Golden [2010] considered the results published in Fischetti et al. [1998].

New optimal solutions have been obtained with the updated execution of the B&C algorithm for all the instances that stopped after 5 hours in Fischetti et al. [1998]: two in generation 1 (ts225, pr226), four in generation 2 (pr266, pr299, lin318, rd400) and four in generation 3 (pr144, pr299, lin318, rd400). The optimal solution for score generation 2 are 6662, 9182, 10923 and 13652, respectively, while in the paper, the mentioned solutions after 5 hours of computation were 6615, 9161, 10900 and 13648, respectively. However, the bounds published for score generation 1 and 3 in the original paper are higher than the values obtained with the updated software. We conjecture that, incidentally, upper bounds were published instead of the best known solutions. For the instances of generation 1, results 125 and 134 appeared in the original paper, and solutions 124 and 126 are now reported, respectively. For the ones of generation 3, old results 3809, 10358, 10382 and 13229 are different from new results 3745, 10343, 10368 and 13223.

The parameters used in the compared heuristic algorithms where those reported by default in the respective papers. However, we have increased two B&C parameters to take advantage of the resources of the current machines. We have experimentally checked that the updated parameters improve the results of the originals parameters. The parameters considered in the runs are as follows:

- B&C: In the cutting plane phase, 200 variables (instead of 100) can be added at each round of pricing up. Additionally, we resort to branching whenever the upper bound did not improve by at least 0.001 in the last 20 (instead of 10) cutting-plane iterations of the current branching node.
- 2-P IA: number of iterations without improvement before termination is 4500, number of nodes to choose from each iteration of route initialization and the number of nodes removed from each iterative change are 4.
- GRASP-PR: greediness parameter is 0.2, number of solutions is 100, constructive methods combine C1 and C2.

Next, the summary results and a comparative analysis is shown for medium-sized instances and large-sized instances. The detailed numerical results can be seen in B.1.

Comparison for medium-sized instances

The TSPLib instances of medium dimensionality contain 45 problems with 48 to 400 nodes. The Table 2.4 summarises the average quality gap (Gap) and time consumption (Time) for the four generations according to the size ranges, the best results between heuristics are highlighted in bold.

Note that all the instances can be solved up to optimality by B&C. However, the execution time is extremely high for this exact approach. In terms of gap, GRASP-PR performed better in generations 1, 3 and 4, while 2-P IA obtained better averages in generation 2, as reported in Tables B.12, B.14, B.16 and B.18.

Comparison with heuristics in medium-sized instances

Taking into account all the medium-sized instances, GRASP-PR obtains the best average gap. In terms of Time, 2-P IA obtains the best results in all the generations. However, EA4OP shows competitive results, obtaining similar execution times to those of 2-P IA in the smallest instances and better time results in the biggest instances.

		B&C		2-P IA		GRASP-PR		EA4OP	
Range	#	Gap	Time	Gap	Time	Gap	Time	Gap	Time
(0,50]	12	*	13.97	0.05	**0.10**	*	0.16	0.06	0.25
(50,100]	56	*	67.24	0.24	**0.36**	**0.10**	0.66	0.25	0.58
(100,150]	44	*	297.23	0.67	**0.77**	**0.38**	2.03	0.67	1.54
(150,200]	24	*	213.34	2.52	**1.90**	1.12	4.40	**0.50**	2.85
(200,250]	20	*	897.83	2.40	**2.45**	1.05	10.06	**0.74**	5.47
(250,300]	16	*	730.80	3.58	4.33	2.66	11.61	**2.18**	**3.71**
(300,350]	4	*	4854.90	3.04	7.46	3.42	19.90	**0.75**	**7.42**
(350,400]	4	*	1429.30	3.80	13.05	4.56	19.35	**1.17**	**7.68**
All	180	*	427.32	1.31	**1.67**	0.80	4.32	**0.63**	2.23

*: optimal solution achieved

Table 2.4: Algorithms comparison by range in medium-sized instances.

Table 2.6 shows the performance of EA4OP versus 2-P IA and GRASP-PR. The table summarizes the following information: *Gap*, number of instances in which an algorithm's solution is higher than the other one's; *Time*, number of instances in which an algorithm's execution time is lower than the other one's; *Pareto*, number of instances in which an algorithm dominates the other algorithm. Pareto efficiency criterion states that a solution dominates the other one if it obtains better results in at least one of the objectives while not degrading any of the others (in our case the objectives are gap and time). Ties are computed in an additional column.

In terms of Gap, EA4OP obtained better solutions than 2-P IA in all four generations, whereas in terms of Time and Pareto, 2-P IA obtains better solutions in all four generations. When we compare EA4OP with GRASP-PR, in terms of Gap and Pareto, EA4OP is better than GRASP-PR in all four generations. In terms of Time, EA4OP obtains better results in generations 1 and 3, and little worse results in generations 2 and 4.

Comparison for large-sized instances

The TSPLib instances of large dimensionality contain 41 problems within 417 and 7397 nodes. Table 2.5 summarises the quality of solutions (Gap) and execution time (Time) for the four generations according to the size ranges. The number of solved instances is detailed in an extra column for B&C and GRASP-PR. The average gap was calculated excluding the missing solutions, whereas the average times were calculated considering 18000 seconds for problems in which the time limit was reached. The best results between heuristics are highlighted in bold.

Most of these instances (130 of 164) can not be solved up to optimality by B&C. Furthermore, B&C finished unexpectedly for 52 of the instances, not obtaining any solution. Globally, EA4OP obtained better solutions than B&C in 96 of the 164 cases.

Comparison with heuristics in large-sized instances

Compared with the rest of the heuristic algorithms, EA4OP obtained better quality solutions in all the generations. Additionally, in this large-sized instance set, EA4OP shows the best performance in execution time compared with the rest of the heuristics in all the generations.

Note that GRASP-PR was not able to return any solution in 22 instances after the execution time was exceeded.

Table 2.7 shows that in large-sized instances EA4OP obtains much better solutions in terms of quality, time and Pareto efficiency, compared with 2-P IA and GRASP-PR for all the generations.

Table 2.5: Comparison of algorithms by range in large-sized instances.

		B&C			2-P IA		GRASP-PR			EA4OP	
Range	#	#	Gap	Time	Gap	Time	#	Gap	Time	Gap	Time
(400,800]	60	55	0.16	9040.89	4.39	**29.15**	60	4.33	224.37	**1.33**	35.95
(800,1200]	20	12	1.24	16910.62	6.09	169.50	20	6.63	1254.55	**1.29**	**67.61**
(1200,1600]	28	15	5.65	16955.71	7.19	352.83	28	6.48	4889.35	**0.57**	**219.35**
(1600,2000]	16	11	15.50	-	4.66	931.49	16	6.27	7271.88	*	**496.68**
(2000,2400]	16	12	10.79	-	5.33	1341.13	15	6.21	8332.97	**0.16**	**733.63**
(2800,3200]	4	3	8.59	-	6.05	3339.27	3	6.60	-	*	**804.96**
(3600,4000]	4	1	10.92	-	7.61	7052.71	0	NA	-	*	**3859.71**
(4400,4800]	4	1	1.16	-	4.86	7527.78	0	NA	-	*	**2455.06**
(5600,6000]	8	1	0.06	-	7.01	15681.39	0	NA	-	*	**5745.60**
(7200,7600]	4	1	25.47	-	15.94	15750.76	0	NA	-	*	**14662.28**
All	164	112	4.21	13343.85	5.73	1899.47	142	5.54	5226.42	**0.76**	**990.42**

*: best known solution achieved

-: time limit of 5 hours exceeded

NA: solution not available after time limit exceeded

Table 2.6: Comparison against state-of-the-art heuristics in terms of quality, time and Pareto efficiency in medium-sized instances.

	Gen1			Gen2			Gen3			Gen4		
	2-P IA	tie	EA4OP	2-P IA	tie	EA4OP	2-P IA	tie	EA4OP	2-P IA	tie	EA4OP
Gap	4	21	20	14	9	22	15	9	21	8	11	26
Time	39	0	6	26	0	19	29	0	16	31	0	14
Pareto	24	0	6	16	0	15	17	0	10	15	0	12
	GRASP-PR	tie	EA4OP	GRASP-PR	tie	EA4OP	GRASP-PR	tie	EA4OP	GRASP-PR	tie	EA4OP
Gap	6	30	9	13	10	22	15	8	22	9	10	26
Time	11	0	34	23	0	22	13	0	32	24	0	21
Pareto	11	0	29	13	0	16	7	0	20	11	0	16

Table 2.7: Comparison against state-of-the-art heuristics in terms of quality, time and Pareto efficiency in large-sized instances.

	Gen1			Gen2			Gen3			Gen4		
	2-P IA	tie	EA4OP	2-P IA	tie	EA4OP	2-P IA	tie	EA4OP	2-P IA	tie	EA4OP
Gap	3	1	37	1	0	40	3	0	38	2	0	39
Time	16	0	25	8	1	32	9	1	31	9	0	32
Pareto	3	0	25	1	0	33	0	0	29	1	0	31
	GRASP-PR	tie	EA4OP	GRASP-PR	tie	EA4OP	GRASP-PR	tie	EA4OP	GRASP-PR	tie	EA4OP
Gap	4	0	37	0	1	40	1	0	40	1	0	40
Time	0	0	41	0	1	40	0	1	40	0	0	41
Pareto	0	0	37	0	0	41	0	0	40	0	0	40

2.5 Conclusions

We have presented an efficient evolutionary algorithm for the OP. Essentially, the algorithm follows the steady-state genetic algorithm schema. It differs in that the proposed method maintains unfeasible solutions during the search and considers specific operators to recover it when required. An Edge Recombination crossover has been adapted for the OP, a novel method for node inclusion has been proposed and the Lin-Kerninghan heuristic has been used to improve route lengths.

The computational experiments have shown that several characteristics are essential in the effectiveness of the EA4OP. Probably the most relevant feature is the use of unfeasible solutions during the search process. It allows us to obtain high quality solutions without being penalized in terms of computational time, as shown in Section 2.4.1. Furthermore, the parameter $d2d$ helps to strike a balance between the solution quality and the computational time.

To our knowledge, the initialization technique of the solutions used in the EA4OP is also novel for the OP. In the proposed initialization, the solutions are built based on the relation between the distance limit and the TSP value of the whole set of nodes (the Lin-Kerninghan approximation of this value). This relation is used to estimate the amount of nodes in the optimal solution and then the solutions are built randomly based on this information. This initialization might be useful, mainly, for population-based algorithms for the variations of the OP to provide diversity to the initial population.

We consider the adaptation of the ER crossover as a contribution to the solution of the OP and routing problems in general. In addition to the problems that consist of permutations, this adaptation also allows us to deal with a wider range of problems whose solution space consists of simple cycles. Moreover, as shown in Section 2.4.2, the proposed crossover turns out to be an effective technique to mix solutions in the OP.

Another contribution that we find remarkable for routing problems is the proposed approach to find the minimum cost insertion in the add operator. When the distance matrix is given by spatial points, its design allows the use of a data structure, i.e., k-d tree, that strongly reduces the computational cost, improving the overall results.

All in all, the EA4OP proves to be an efficient algorithm for the OP. Not only does the EA4OP obtain competitive results in medium-sized instances in comparison to the state-of-the-art algorithms, but it also achieves outstanding results in terms of quality in an even lower execution time.

We have tested the EA4OP in 344 instances based on TSPLib. We have found the EA4OP competitive in medium-sized instances (up to 400 nodes). Comparing the EA4OP in terms of Pareto efficiency, we have found that from the 180 instances of the medium-sized set, EA4OP gets 43 Pareto optimums while 2-P IA does so for 72 instances. Also, EA4OP obtains 81 Pareto optimums, while the GRASP-PR does so for 42 instances. As for the medium-sized instances, B&C has been run again with an updated LP solver in a modern machine, and 10 new optimal solutions were found: two in generation 1, four in generation 2 and four in generation 3 (for these instances, execution in Fischetti et al. [1998] stopped because the time limit was reached).

The computational results on large-sized instances (up to 7397 nodes) are excellent for the EA4OP in terms of quality and time. Moreover, the EA4OP algorithm found higher solutions than the ones returned by the exact approach after five hours of computation. Additionally, the execution time is lower than the ones of the rest of the compared techniques. Particularly, from the 164 instances of the large-sized set, EA4OP obtained the Pareto optimum in 118 instances, while the 2-P IA, which turns out to be the most competitive heuristic algorithm, did it for 5 instances.

Ordering the algorithms in terms of average quality gap, we have obtained the following results: for medium-sized instances, B&C (0.00%), EA4OP (0.63%), GRASP-PR (0.80%) and 2-P IA (1.31%); and for large-sized instances, EA4OP (0.76%), B&C (4.21%), 2-P IA (5.73%) and GRASP-PR (5.54%).

Ordering the algorithms in terms of average time consumption, we have obtained the following results: for medium-sized instances, 2-P IA (1.67 sec), EA4OP (2.23 sec), GRASP-PR (4.32 sec) and B&C (427.32 sec); and for large-sized instances, EA4OP (990.42 sec), 2-P IA (1899.47 sec), GRASP-PR (5226.42 sec) and B&C (13343.85 sec).

In order to obtain better quality solutions or decrease time consumption, it would be interesting to advance developing new operators or adapt the ones developed for other routing problems. Additionally, it could be revelant to build better quality initial populations. Giving a different a priori probability to each node might contribute to this aim. Furthermore, it would be challenging to consider the very large-sized instances, in particular, 26 TSPLib instances left with nodes from 11849 to 85900. Another point of particular interest would be the application of the EA4OP to solve classical variants of OP (such as the team OP, the OP with time windows or the time dependent OP) as well as recent ones (such as the stochastic OP, the generalized OP, the arc OP, the multi-agent OP or the clustered OP).

CHAPTER 3

Shrinking and Separation Algorithms for Cycle Problems

OUTLINE

In this chapter, we study the shrinking of support graphs and the exact algorithms for subcycle elimination separation problems. The efficient application of the considered techniques has proved to be essential in the solution of large-sized Travelling Salesperson Problem, and this has been the motivation behind this work. Regarding the shrinking of support graphs, we prove the validity of the Padberg-Rinaldi general shrinking rules and the Crowder-Padberg subcycle-safe shrinking rules. Concerning the subcycle separation problems, we extend two exact separation algorithms, the Dynamic Hong and the Extended Padberg-Grötschel algorithms, which are shown to be superior to the ones used so far in the literature of cycle problems.

3.1 Introduction

The Travelling Salesperson Problem (TSP) has been the source and the testbed of the most important techniques developed for the exact solution of combinational optimization problems. These techniques have been principally developed in the context of the Branch-and-Cut (B&C) algorithm, which combines the Branch-and-Bound (B&B) and the cutting-planes methods, see Applegate et al. [2007] for an historical overview. Eventually, many of these techniques have been successfully adapted to other related problems. However, there are procedures, such as the support graph shrinking and some separation algorithms, that are strongly dependent on the problem peculiarities. As a consequence, these techniques might not have been adapted yet, or there might still be room for further improvements.

As TSP is the most well-known cycle problem, we motivate the goals of this chapter focusing on this problem. When a B&B algorithm is used to exactly solve the TSP, which is an Integer Problem (IP), the cutting-planes method arises as a natural strategy to handle at least two situations: the exponential number of constraints of the model and the consequences of the linear relaxation of the integer problem. Recall that in a B&B algorithm the branching decisions are made guided by a sequence of Linear Problems (LP). These LPs are principally obtained by relaxing the integrality and fixing the variables according to the preceding branching decisions.

Within this approach, the cutting-planes method is required due to the fact that, in order to define a TSP model, an exponential number of constraints in terms of the number of vertices in the TSP is needed, see Padberg and Sung [1991]. In order to deal with this situation, the exact algorithm is initialized with a subproblem of the LP, let us call this LP_0, that considers a controlled number of constraints. During the algorithm, the excluded constraints are added to LP_0 only if they are required, i.e., if they are violated by the solution of the LP_0. The second reason to consider the cutting-planes method is that since the variables in the linear relaxation of the TSP are considered continuous instead of integers, new families of valid inequalities arise (inequalities that are satisfied by all the cycles), also called cuts, that are not linear combinations of the constraints defining the TSP. Since the number of branch nodes needed to visit by the algorithm is reduced, the cutting-planes are very valuable to decrease the solving time of a B&B algorithm.

Computationally, the most expensive part of the cutting-planes method is to solve the separation problems. Given a solution of the LP_0 and an inequality family, the separation problem for the given family consists of finding either the violated inequalities of the family or a certificate that no violated inequality of the family exists.

Motivation

The difficulty of efficiently solving the separation problems becomes evident when the number of vertices of the problem increases. It is well known that, in practice, even a polynomial time separation algorithm might turn out to be inefficient for certain families. To mitigate this issue, a technique known as shrinking has been exploited in the TSP, see Crowder and Padberg [1980]; Padberg and Rinaldi [1990b]; Grötschel and Holland [1991]. Shrinking consists of safely simplifying, i.e., without losing all the violated inequalities of the family, the support graph generated by the solution of the LP_0. This way, considering that, generally, the separation is harder than the shrinking, the cost of finding the violated inequalities is reduced because the separation is performed in a graph involving a lower number of vertices and edges.

In Figure 3.1, a flowchart of a generic B&C algorithm and the separation algorithm with and without the shrinking.

In the last few decades, many optimization problems have proliferated whose solution is required to be a cycle, but not necessarily Hamiltonian as in the TSP. This is the case for some extensions of the TSP itself, as can be seen in the extensive collection about TSP variants of Gutin and Punnen [2007]. For instance, the weighted girth problem, consists of finding the minimum cost cycle in a weighted graph, see Coullard and Pulleyblank

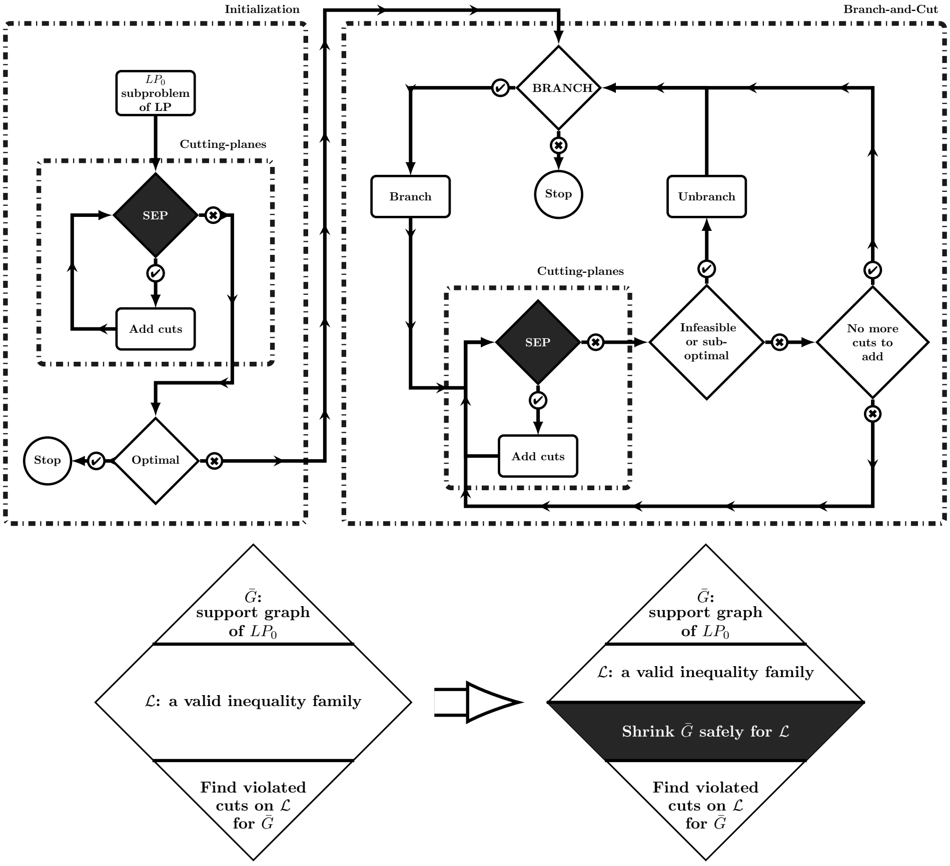


Figure 3.1: In the top, a flowchart of a generic Branch-and-Cut algorithm. BRANCH is an oracle which returns an unevaluated node in the branching tree. At each action box of the flowchart the subproblem LP_0 is updated and solved. In the bottom, the detailed separation algorithm (SEP) without and with shrinking.

[1989] and Bauer [1997]. Cycles are also the solutions of the Generalized TSP (GTSP) where the vertices are labeled in clusters and at least one vertex of each cluster is required to be visited, but not all the vertices, see Fischetti et al. [1995]. Other routing problems, which are recently gaining popularity because of their wide range of applications, are the TSP with profits, see Feillet et al. [2005] and Archetti et al. [2014b]. These problems are the Profitable Tour Problem (PTP), the Orienteering Problem (OP), the Price Collecting TSP (PCTSP), and their variations. From the TSP with profits, the OP, which consists of finding the cycle that maximizes the collected vertex profits subject to a cycle length constraint, is the one which has been most extensively studied. For a recent book on applications and variants of the OP see Vansteenwegen and Gunawan [2019].

This chapter has three main aims: first, to generalize the shrinking rules (global and subcycle specific) proposed in the literature of the TSP to the case of cycle problems; second, to extend in an effective manner the subcycle exact separation algorithms for cycle problems; and third, to show experimentally the relevance of the proposed shrinking rules and separation algorithms. On the one hand, 6 different shrinking rules for cycle problems are presented in this work, of which three are safe for all the valid inequalities and three are specifically safe for subcycle elimination constraints. On the other hand, we extend two exact separation algorithms proposed in Padberg and Grötschel [1985] and Padberg and Rinaldi [1990b]. We empirically show the contribution of the shrinking and separation strategies in the time reduction and in the generation of violated subcycle elimination constraints. For the experiments, we have used 24 instances of the subcycle separation problem generated in the solution of OP by B&C with up to 15112 number of vertices. The results show that the speedup of using the combination of the proposed shrinking and separation techniques is around 50 times in medium-sized instances and 200 times in large-sized instances.

3.2 The Cycle Polytope

Let $G = (V, E)$ be an undirected graph with no loops. Let us define the following sets:

$$(Q : W) := \{[u, v] \in E : u \in Q, v \in W\} \qquad Q, W \subset V \tag{3.1a}$$
$$\delta(Q) := (Q : V - Q) \qquad Q \subset V \tag{3.1b}$$
$$E(Q) := (Q : Q) \qquad Q \subset V \tag{3.1c}$$
$$V(T) := \{v \in V : T \cap (v : V) \neq \emptyset\} \qquad T \subset E \tag{3.1d}$$
$$N(Q) := V(\delta(Q)) - Q \qquad Q \subset V \tag{3.1e}$$

where $(Q : W)$ are the edges connecting Q and W, $\delta(Q)$ is the set of edges in the coboundary of Q also known as the star-set of Q, $E(Q)$ is the set of edges between the vertices of Q, $V(T)$ is the set of vertices incident with an edge set T, and $N(Q)$ are the neighbour vertices set of Q. For simplicity, we sometimes denote $\{e\}$ and $\{v\}$ by e and v, respectively, e.g., $\delta(v)$ and $V(e)$.

We denote by $\mathbb{R}^V$ and $\mathbb{R}^E$ the space of real vectors whose components are indexed by elements of V and E, respectively. With every subset $T \subset E$ we associate a vector $(y, x)^T = (y^T, x^T)$ called the characteristic vector of T, defined as follows:

$$y_v^T := \begin{cases} 1 & \text{if } v \in V(T) \\ 0 & \text{otherwise} \end{cases} \qquad x_e^T := \begin{cases} 1 & \text{if } e \in T \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

When $y_v^T = 1$, i.e. $v \in V(T)$, we say that the vertex v is visited by the edge set T.

We denote by $\mathcal{C}_G$ the set of (simple) cycles of the graph G. We assume that every cycle $\tau \in \mathcal{C}_G$ is represented as a subset of edges. Then, the cycle polytope P_C^G of the graph G is the convex hull of the characteristic vectors of all the cycles of the graph:

$$P_C^G := conv\{(y, x)^\tau \in \mathbb{R}^{V \times E} : \tau \in \mathcal{C}_G\} \tag{3.3}$$

By definition, a vector (y, x) belongs to P_C^G if it is a convex combination of cycles of $\mathcal{C}_G$, i.e., $(y, x) \in P_C^G$ if and only if there exists a set of real numbers $\{\lambda_\tau\}_{\tau \in \mathcal{C}_G}$ such that

$$(y, x) = \sum_{\tau \in \mathcal{C}_G} \lambda_\tau (y, x)^\tau \tag{3.4}$$

$\lambda_\tau \geq 0$ for every $\tau \in \mathcal{C}_G$ and $\sum_{\tau \in \mathcal{C}_G} \lambda_\tau = 1$.

Similarly, we denote by $\mathcal{T}_G$ the set of tours, i.e., Hamiltonian cycles, of the graph G, and by P_{TSP}^G the TSP polytope of the graph G. The P_{TSP}^G is the convex hull of the characteristic vectors of all the tours of the graph:

$$P_{TSP}^G := conv\{(y, x)^\tau \in \mathbb{R}^{V \times E} : \tau \in \mathcal{T}_G\} \tag{3.5}$$

Note that, $y = 1$ is satisfied by every $(y, x) \in P_{TSP}^G$. Since, the tours form a subset of cycles of G, we have that:

$$P_{TSP}^G \subset P_C^G \tag{3.6}$$

In order to use Linear Programming based techniques such as the B&C algorithm, the polytope P_C^G must be characterized by means of a system of linear constraints. A complete characterization of the integer points of P_C^G using only edge variables was given in Bauer [1997]. In this work, since we find it more convenient to formulate the shrinking rules of Section 3.3.1 and Section 3.3.2, we consider an equivalent one which uses the vertex and edge variables for the characterization. For $(y, x) \in \mathbb{R}^{V \times E}$, $S \subset V$ and $T \subset E$, we define $y(S) = \sum_{v \in S} y_v$ and $x(T) = \sum_{e \in T} x_e$. Let us consider the following constraints:

$$x(\delta(v)) - 2y_v = 0, \qquad v \in V \tag{3.7a}$$

$$y_v - x_e \geq 0, \qquad \forall v \in V,\ e \in \delta(v) \tag{3.7b}$$

$$x(\delta(Q)) - 2y_v - 2y_w \geq -2, \qquad v \in Q \subset V, 3 \leq |Q| \leq |V| - 3, \quad w \in V - Q \tag{3.7c}$$

$$x(E) \geq 3, \tag{3.7d}$$

$$1 \geq y_v \geq 0, \qquad \forall v \in V \tag{3.7e}$$

$$x_e \geq 0, \qquad \forall e \in E \tag{3.7f}$$

$$x_e \in \mathbb{Z} \qquad \forall e \in E \tag{3.7g}$$

The degree equations (3.7a) together with the logical constraints (3.7b) and the integrality constraints (3.7g) ensure that the visited vertices have exactly two incident edges and the unvisited vertices none. The Subcycle Elimination Constraints (SEC) (3.7c) ensure that only one connected cycle exists. Throughout the thesis, we use the notation $\langle Q, v, w \rangle$ to refer to the SEC defined by the set Q and the vertices $v \in Q$ and $w \notin Q$. In the literature, the SECs have also been called Generalized Subtour Elimination Constraints (GSEC). The inequality (3.7d) imposes the property that the undirected cycles contain at least 3 edges. The conditions (3.7e), (3.7f) and (3.7g) impose that all the variables are 0-1. Note that the integrality of the y_v variables is ensured by (3.7a), (3.7b) and (3.7g), and the condition $x_e \leq 1$ is ensured by (3.7b) and (3.7e). Considering the constraints in (3.7), the cycle polytope of a graph $G = (V, E)$ can be expressed as follows:

$$P_C^G = conv\{(y, x) \in \mathbb{R}^{V \times E} : (y, x) \text{ satisfies } (3.7a), (3.7b), (3.7c), (3.7d), (3.7e), (3.7f), (3.7g)\} \quad (3.8)$$

In some problems, for instance OP and PCTSP, a feasible solution must visit a depot vertex, i.e., $y_d = 1$ for a vertex $d \in V$. In such cases, the family of SECs (3.7c) that define the cycle polytope can be substituted with the following subfamily:

$$x(\delta(Q)) - 2y_v \geq 0, \qquad v \in Q \subset V, 3 \leq |Q| \leq |V| - 3,\ d \notin Q \quad (3.9)$$

where each constraint can be represented as $\langle Q, v \rangle$. In a B&C algorithm, where all the constraints of the model are not considered in the LP_0, the only advantage by using this constraint family is that we simplify a vertex in the SEC representation. However, it has one important disadvantage, in the family (3.9) we might need to consider an SEC with $|Q| > |V|/2$, while in the family (3.7c) it can be considered always a SEC such that $|Q| \leq |V|/2$. Therefore, we always consider the family (3.7c) regardless of whether it is given a depot or not in the cycle problem.

When a B&C is used to solve a cycle problem, the integrality constraints (3.7g) of the P_C^G are relaxed in order to first seek a solution that satisfies the rest of the constraints. Contrary to this strategy, Pferschy and Staněk [2017] have recently considered again relaxing the SEC constraints in the TSP, to first solve the resulting problem to integer optimality with MILP-solvers and then introduce the SECs if required. Despite the improvement of the new MILP-solvers, this approach is still inferior compared to the opposite strategy. As a consequence of the continuous relaxation, a solution (y, x) that satisfies the rest of the constraints of (3.7) might still not belong to P_C^G. In these cases, instead of directly resorting to the branching phase to tighten the integrality gap, we could check if additional (not dominated by those in (3.7)) and facet-defining valid inequalities for the P_C^G are violated. The strength of considering additional valid inequalities was shown in the 1970s in the study of the TSP Grötschel and Padberg [1979]. In Bauer [1997] an extension of the clique trees inequality family (originally defined for the TSP) was given, which includes the so-called comb inequalities, for cycle problems. The shrinking rules proposed in Section 3.3.1 are safe for all the valid inequalities for P_C^G.

A polytope that it is closely related to P_C^G is the so-called lower cycle polytope, see Bauer [1997]:

$$L_C^G = conv\{P_C^G, (0, 0)\} \quad (3.10)$$

where $(0, 0) \in \mathbb{R}^{V \times E}$ is the vector that represents that no vertex and edges of the graph are visited. It is easy to see, that for every graph G, so that it contains at least one cycle, there exist an infinity number of vectors $(y, x) \in L_C$ such that $x(E) < 3$. Hence, the polytope P_C^G is a proper subspace of L_C^G for every graph G that contains at least one cycle. It is crucial to consider the polytope L_C^G to obtain the shrinking results in Section 3.3.1.

In a B&C algorithm, it is reasonable to solve the separation problems of the valid inequality families following an order determined by their complexity. This order defines a hierarchy of the inequality families and their closure polytopes. We refer to the closure polytope of an inequality family as the polytope that satisfies all the inequalities of the given family and its preceding families in this hierarchy.

Without considering the variable bounds (3.7e)-(3.7f) and the inequality (3.7d), the simplest inequalities are the degree equations (3.7a) and the logical constraints (3.7b). These have, respectively, linear and quadratic exact algorithms in terms of the number of the vertices of G and generally are always included in the LP_0. The closure polytope of the inequalities (3.7a) and (3.7b) (the inequality (3.7d) is excluded to favour the convexity) turns out to be the undirected Assignment Polytope (with loops), P_A^G, which is defined as:

$$P_A^G := \{(y, x) \in \mathbb{R}^{V \times E} : (y, x) \text{ satisfies (3.7a), (3.7b), (3.7e), (3.7f)}\} \tag{3.11}$$

Next in the hierarchy comes the SEC family. A straightforward exact separation algorithm for the SECs has $O(|V|^4)$ time complexity (see Section 3.5.3 for further discussion) and its closure polytope is defined as:

$$P_{SEC}^G := \{(y, x) \in P_A^G : (y, x) \text{ satisfies (3.7c)}\} \tag{3.12}$$

Considering the relationship $P_C^G \subset P_{SEC}^G \subset P_A^G$, the underlying purpose of this chapter is to effectively determine if a given solution $(y, x) \in P_A^G$ of a LP_0 belongs to P_{SEC}^G, or in case that it does not belong, to provide the violated inequalities.

Throughout the chapter, we make use of the following well-known identity repeatedly. Given a graph G, a subset $S \subset V$ and a vector $x \in \mathbb{R}^E$, the identity

$$x(\delta(S)) = \sum_{v \in S} x(\delta(v)) - 2x(E(S)) \tag{3.13}$$

is always satisfied. In addition, if the vector $(y, x) \in \mathbb{R}^{V \times E}$ satisfies the degree constraints (3.7a), then the equations

$$x(\delta(S)) = 2y(S) - 2x(E(S)) \quad S \subset V \tag{3.14}$$

are satisfied by the vector (y, x). Particularly, the identity (3.14) is satisfied by every vector in P_{TSP}^G, P_C^G, P_{SEC}^G and P_A^G.

Let $G^* = (V^*, E^*)$ be the support graph of a given vector (y, x) where

$$V^* := \{v \in V : y_v > 0\} \tag{3.15a}$$

$$E^* := \{e \in E : x_e > 0\} \tag{3.15b}$$

Figure 3.2 shows a support graph obtained when solving the instance pr76 (TSPLIB) for Generation 1 with B&C, while Figure 3.2 shows its topological representation. In the figures, the vertices and the edges with value 1 are represented in black. The vertices and the edges with value in $[0.5, 1)$ are represented in red. The vertices in white and the edges with dashed style represent those with value in $(0, 0.5)$. The edges in blue and double lined style represent those with value greater than 1. The depot vertex of the OP, the vertex 1, is colored in green.

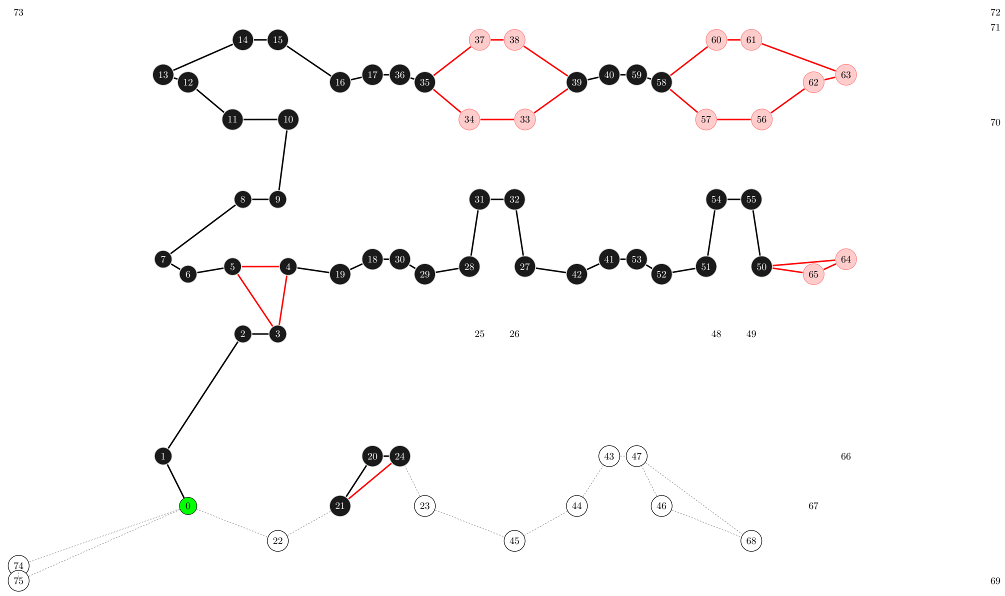


Figure 3.2: Example of a support graph obtained when solving instance pr76-gen1 by Branch-and-Cut.

3.3 Shrinking for the Cycle Polytope

Let us introduce the following notation. Given a graph $G = (V, E)$, the vector $(y, x) \in \mathbb{R}^{V \times E}$ and a subset $S \subset V$, we denote by $G[S] = (V[S], E[S])$ the graph obtained by shrinking the set S into a single vertex $s \notin V$, where the resulting set of vertices and edges are as follows:

$$V[S] = (V - S) \cup \{s\} \tag{3.16a}$$
$$E[S] = E(V - S) \cup \{[s, v] : v \in V - S, x(S : v) > 0\} \tag{3.16b}$$

and by $(y[S], x[S]) \in \mathbb{R}^{V[S] \times E[S]}$ we denote the vector with components

$$x[S]([u, v]) = x_{[u,v]} \qquad \forall [u, v] \in E \cap E[S] \tag{3.17a}$$
$$x[S]([s, v]) = x(S : v) \qquad \forall v \in V - S \tag{3.17b}$$
$$y[S](v) = y_v \qquad \forall v \in V \cap V[S] \tag{3.17c}$$

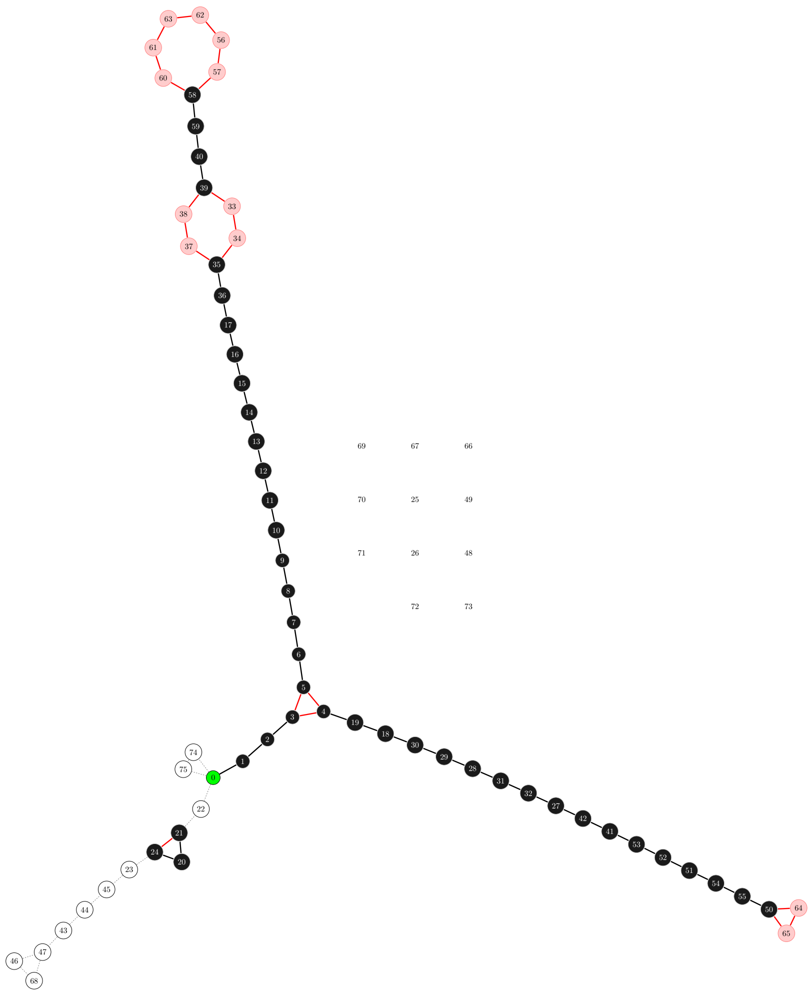


Figure 3.3: Topological representation of the support graph in Figure 3.2

$$y[S](s) = x(\delta(S))/2 \tag{3.17d}$$

Let $Q \subset V$ be a subset of vertices, we denote with $Q[S]$ the subset derived by shrinking S

$$Q[S] = \begin{cases} (Q - S) \cup \{s\} & \text{if } S \cap Q \neq \emptyset \\ Q & \text{otherwise} \end{cases} \tag{3.18}$$

which has the following associated values:

$$y[S](Q[S]) = \begin{cases} y(Q) - y(Q \cap S) + \dfrac{x(\delta(S))}{2} & \text{if } S \cap Q \neq \emptyset \\ y(Q) & \text{otherwise} \end{cases} \tag{3.19a}$$

$$x[S](\delta(Q[S])) = \begin{cases} x(\delta(S \cup Q)) & \text{if } S \cap Q \neq \emptyset \\ x(\delta(Q)) & \text{otherwise} \end{cases} \tag{3.19b}$$

$$x[S](E(Q[S])) = x(E(Q)) - x(E(Q \cap S)) \tag{3.19c}$$

3.3.1 Shrinking for the Cycle Polytope

In this section, we present three shrinking rules that are safe for the P_C^G. In essence, we have generalized for every (simple) cycle problem the results obtained by [Padberg and Rinaldi, 1990b] for Hamiltonian cycle problems. In the following lines, we formalize the concept of safe shrink for P_C^G and we prove the lemmas and the theorem in which shrinking rules for cycle problems are based on. In addition, we show that the three shrinking rules can be consecutively applied for the P_C^G.

Based on the definition given in [Padberg and Rinaldi, 1990b] for safe shrinking for the P_{TSP}^G, an analogue definition can be formulated for safe shrinking for the P_C^G.

Definition 3.1. *Given a vector $(y, x) \notin P_C^G$, a set $S \subset V$ is safe to shrink if $(y[S], x[S]) \notin P_C^{G[S]}$.*

Note that the definition does not assume a one-by-one relationship between the violated inequalities of (y, x) and $(y[S], x[S])$. A set S that is safe to shrink for a separable solution (y, x) from P_C^G should be understood as a subset when shrinking it does not project the solution (y, x) to $P_C^{G[S]}$. When a set S is safe to shrink for a given (y, x), it is also said that S is shrinkable for (y, x).

The definition of shrinkable set does not provide a practical tool for finding them. Hence, the first goal is to give a set of rules of shrinking for P_C^G, which are obtained in Theorem 3.5. The strategy used in [Padberg and Rinaldi, 1990b] to obtain the shrinking rules for tours cannot be applied directly for simple cycles, because it relies on the fact that the tours visit every vertex in the graph. So, first we need to obtain the following lemma.

Lemma 3.1. *Let $(y, x) \in L_C^G$ be a vector. Suppose that $\{Q, \{u\}, \{v\}\}$ is a partition of V such that $x_{[u,v]} = x(u : Q) = x(v : Q) > 0$. Then any cycle τ of $\mathcal{C}^G$ that has a positive coefficient in the convex combination of (y, x), $\lambda_\tau > 0$, fulfills one of the following cases:*

(i) $V(\tau) \subset Q$

(ii) $|\tau \cap (u : Q)| = |\tau \cap (v : Q)| = |\tau \cap [u, v]| = 1$

Proof. Let $\mathcal{C}_{uv}$ denote the subset of cycles in $\mathcal{C}$ that visits the edge $[u, v]$ and has a positive value, $\lambda_\tau > 0$. Note that since $(y, x) \in L_C^G$, then $x_{[u,v]} \leq y_v$ and $x_{[u,v]} \leq y_u$. So, in order to satisfy the degree equations, every cycle τ in $\mathcal{C}_{uv}$ must contain at least an edge in $(u : Q)$ and $(v : Q)$. Moreover, since τ is a simple cycle, every $\tau \in \mathcal{C}_{uv}$ crosses exactly once $(u : Q)$ and $(v : Q)$. Now, let us see that if τ does not belong to $\mathcal{C}_{uv}$ and $\lambda_\tau > 0$, then τ is contained in Q. Consider the following inequality:

$$x_{[u,v]} = \sum_{\zeta \in \mathcal{C}_{uv}} \lambda_\zeta x^\zeta_{[u,v]} = \sum_{\zeta \in \mathcal{C}_{uv}} \lambda_\zeta = \sum_{\zeta \in \mathcal{C}_{uv}} \sum_{e \in (u:Q)} \lambda_\zeta x^\zeta_e \leq \tag{3.20a}$$

$$\sum_{\zeta \in \mathcal{C}_{uv}} \sum_{e \in (u:Q)} \lambda_\zeta x^\zeta_e + \sum_{\zeta \notin \mathcal{C}_{uv}} \sum_{e \in (u:Q)} \lambda_\zeta x^\zeta_e = x(u : Q) \tag{3.20b}$$

Since $x_{[u,v]} = x(u : Q)$, we have that $x^\tau_e = 0$ for every $e \in (u : Q)$. Similarly, we obtain that $x^\tau_e = 0$ for every $e \in (v : Q)$. Therefore, τ is contained in Q. □

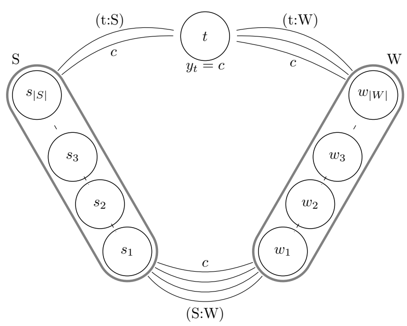


Figure 3.4: Illustration of the scenario in Lemma 3.2.

The next result generalizes the main theorem of shrinking in [Padberg and Rinaldi, 1990b]. The principal idea is to use a constant, c, to extend the rules of the original paper (where $\forall v \in V$ satisfies $y_v = 1$) for vertices that have fractional value. We also need an additional hypothesis about the vector $(y[W], x[W])$ obtained by shrinking the subset W, the "complement" of S, which is not required for the TSP because it is trivially satisfied by Hamiltonian cycles.

Lemma 3.2. *Given a vector $(y, x) \notin P^G_C$, let $\{S, W, \{t\}\}$ be a partition of V with $2 \leq |S|$ and c be a constant where $0 < c \leq 1$ such that:*

(i) $y_v = c \ \forall v \in S \cup \{t\}$

(ii) $x(E(S)) = c \cdot (|S| - 1)$

(iii) $x(t : S) = c$

(iv) $(y[W], x[W]) \in L^{G[W]}_C$

(v) No cycle in the convex combination of $(y[W], x[W])$ is contained in S

Then it is safe to shrink S for (y, x).

Proof. Based on the hypotheses i), ii) and iii) of the lemma and the identity (3.14) we obtain that $x(S : W) = c$ and $x(t : W) = c$, as illustrated in Figure 3.4.

Suppose for contradiction that S is not shrinkable, so $(y[S], x[S]) \in P_C^{G[S]}$. Since $x_{[s,t]} = x(s : W) = x(t : W)$, based on Lemma 3.1, the vector $(y[S], x[S])$ can be written as:

$$(y[S], x[S]) = \sum_{\zeta \in \mathcal{W}_s} \alpha_\zeta (y, x)^\zeta + \sum_{\zeta \in \mathcal{W}_0} \alpha_\zeta^0 (y, x)^\zeta \tag{3.21}$$

where $\mathcal{W}_s$ is the set of cycles visiting the shrunk vertex s having $\alpha_\zeta > 0$ and $\mathcal{W}_0$ is the set of cycles contained in W having $\alpha_\zeta^0 > 0$. Note that $\mathcal{W}_0$ might be an empty set. The coefficients satisfy $\sum_{\zeta \in \mathcal{W}_s} \alpha_\zeta + \sum_{\zeta \in \mathcal{W}_0} \alpha_\zeta^0 = 1$.

By hypothesis the vector $(y[W], x[W])$ belongs to $L_C^{G[W]}$, so $(y[W], x[W])$ can be written as a convex combination of cycles of $\mathcal{C}_{G[W]}$ and the vector $(0, 0)$. Because of the Lemma 3.1 and by the hypothesis v) the vector $(y[W], x[W])$ can be written as:

$$(y[W], x[W]) = \sum_{\eta \in \mathcal{S}_w} \beta_\eta (y, x)^\eta + \beta_{(0,0)}(0, 0) \tag{3.22}$$

where $\mathcal{S}_w$ is the set of cycles visiting w (the vertex to which W is contracted to) having $\beta_\eta > 0$, $\beta_{(0,0)} \geq 0$ and $\sum_{\eta \in \mathcal{S}_w} \beta_\eta + \beta_{(0,0)} = 1$.

Now, considering $x(t : s) = x(t : w) = c$ we have that:

$$c = \sum_{\zeta \in \mathcal{W}_s} \alpha_\zeta = \sum_{\eta \in \mathcal{S}_w} \beta_\eta \tag{3.23}$$

and from the fact that the coefficients sum up to one, we have that:

$$1 - c = \sum_{\eta \in \mathcal{W}_0} \alpha_\eta^0 = \beta_{(0,0)} \tag{3.24}$$

To prove the lemma we follow the “patch-and-weight” strategy used in [Padberg and Rinaldi, 1990b] for the P_{TSP}^G whose goal is to reconstruct the cycles and coefficients of the convex combination of the vector (y, x). According to the vertices in W, we can partition $\mathcal{W}_s$ into $|W|$ pairwise disjoint subsets (some of them which be empty). For $j \in \{1, \ldots, |W|\}$ let us call $\mathcal{W}_s^j$ the subset of cycles in $\mathcal{W}_s$ containing the edge $[s, w_j]$, and denote by $\zeta_1^j, \ldots, \zeta_{k_j}^j$ the cycles of $\mathcal{W}_s^j$ and by $\beta_1^j, \ldots, \beta_{k_j}^j$ their coefficients in the convex combination. In the same way, we can partition $\mathcal{S}_w$ into $|S|$ subsets calling $\mathcal{S}_w^i$ the subset of cycles in $\mathcal{S}_w$ containing the edge $[s_i, w]$. We denote by $\eta_1^i, \ldots, \eta_{h_i}^i$ the cycles of $\mathcal{S}_w^i$ and by $\alpha_1^i, \ldots, \alpha_{h_i}^i$ their coefficients in the convex combination.

The cycles of the convex combination of (y, x) are constructed in two steps. In the first step, $|\mathcal{S}_w|$ copies of each cycle in $\mathcal{W}_s$ are created. With this goal, for each $j \in \{1, \ldots, |W|\}$ and for each $l \in \{1, \ldots, k_j\}$, create $|S|$ copies of the cycle ζ_l^j, and denote them by $\{\tau_l^{ij}\}$ for $i \in \{1, \ldots, |S|\}$. Then, for each $j \in \{1, \ldots, |W|\}$, for each $l \in \{1, \ldots, k_j\}$ and for each $i \in \{1, \ldots, |S|\}$ create h_i copies of τ_l^{ij}, and denote them by $\{\tau_{ml}^{ij}\}$ for $m \in \{1, \ldots, h_i\}$. At this point we have $|\mathcal{W}_s| \cdot |\mathcal{S}_w|$ cycles that belong to $G[S]$. In the second step, these cycles of $G[S]$ are extended to cycles of G. To that end, consider each cycle τ_{ml}^{ij} and remove the edges $[t,s]$ and $[s,w_j]$ and join the resulting path with the path in $G[W]$ obtained from the cycle η_m^i by removing the edges $[w,t]$ and $[s_i,w]$, and add the edge $[s_i,w_j]$ to obtain the extension of τ_{ml}^{ij} to G.

The coefficients of the constructed τ_{ml}^{ij} cycles are defined in the following way:

$$\lambda_{ml}^{ij} = \frac{x_{[s_j,w_i]} \cdot \alpha_l^j \cdot \beta_m^i}{\sum_{r=1}^{k_j} \alpha_r^j \cdot \sum_{r=1}^{h_i} \beta_r^i} \tag{3.25}$$

where $i \in \{1,\dots,|S|\}$, $j \in \{1,\dots,|W|\}$, $m \in \{1,\dots,h_i\}$ and $l \in \{1,\dots,k_j\}$. It can be verified that the coefficients defined this way sum c in total:

$$\sum_{i,j,m,l} \lambda_{ml}^{ij} = \sum_{i,j} x_{[s_j,w_i]} \sum_{m,l} \frac{\alpha_l^j \cdot \beta_m^i}{\sum_{r=1}^{k_j} \alpha_r^j \cdot \sum_{r=1}^{h_i} \beta_r^i} \tag{3.26a}$$

$$= \sum_{i,j} x_{[s_j,w_i]} = x(S:W) = c \tag{3.26b}$$

Then the vector (y,x) can be obtained as a convex combination of the cycles in $\mathcal{W}_0$ and $\{\tau_{ml}^{ij}\}$ with coefficients $\{\alpha_\varsigma^0\}$ and $\{\lambda_{ml}^{ij}\}$, respectively. We conclude $(y,x) \in P_C^G$ which is a contradiction. □

The lemma gives a sufficient condition for a set to be shrinkable, but still it is not practical. The next theorem gives three practical scenarios to make use of Lemma 3.2. Beforehand, let us obtain a useful result for L_C^G. Consider the undirected version of the Assignment Polytope (without loops) P_A^1 defined as:

$$P_A^1 := \{(y,x) \in \mathbb{R}^{V\times E} : (y,x) \text{ satisfies (3.7a), (3.7b), (3.7f)}, y = 1\} \tag{3.27}$$

It is a well-known result of the literature that $P_{TSP}^G = P_A^1$ for $3 \le |V| \le 5$ (see [Grötschel and Padberg, 1979]). This relationship is the key to obtaining the shrinking rules for the P_{TSP}^G in [Padberg and Rinaldi, 1990b]. So, we would like to obtain a similar result for L_C^G and P_A^G. However, $L_C^G \neq P_A$ when $4 \le |V|$, as shown in the counterexample of Figure 3.5. The vector defined in the figure belongs to P_A^G, but it does not belong to L_C^G, because it cannot be expressed as a convex combination of cycles.

Nevertheless, we have the following lemma which is enough to prove Theorem 3.5.

Lemma 3.3. *Let $G = (V,E)$ be a graph and c be a constant such that $3 \le |V| \le 5$ and $0 < c \le 1$. If $(y,x) \in P_A$ such that $y_v = c$ for all $v \in V$, then $(y,x) \in L_C$.*

Proof. It is straightforward that if $(y,x) \in P_A$ such that $y_v = c$ for all $v \in V$, then $\frac{1}{c}(y,x) \in P_A^1$. By the classical result in [Grötschel and Padberg, 1979], since $3 \le |V| \le 5$, the equality $P_A^1 = P_{TSP}$ is satisfied. Since P_{TSP}^G is contained in L_C^G, the vector $\frac{1}{c}(y,x)$ belongs to L_C^G. Then, since both $(0,0)$ and $\frac{1}{c}(y,x)$ belong to L_C^G, which is convex, and $0 \le c \le 1$ we have that $(y,x) \in L_C$. □

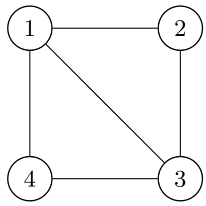


Figure 3.5: An example of a solution that belongs to P_A^G but not to L_C^G when $|V| = 4$ (it can be easily extended for $|V| \geq 4$ by means of subdivisions). All the edges in the figure have value $\frac{1}{2}$. The values of the vertices satisfy the degree equations.

Lemma 3.4. *Given a graph G such that $|V| = 5$, a vector $(y, x) \in L_C^G$ and $0 \leq c \leq 1$, suppose that $\lambda_{(0,0)} = 1 - c$. Let $\{S, \{t\}, \{w\}\}$ be a partition of V such that $x_{[t,w]} = x(t : S) = x(w : S) = c$, then every cycle τ in $\mathcal{C}^G$ such that $\lambda_\tau > 0$ is not contained in S.*

Proof. Since $\{S, \{t\}, \{w\}\}$ is a partition of V, we have that $|S| = 3$ and $|V - S| = 2$. Hence, every cycle in $\mathcal{C}^G$ has vertices in S. According to the number of visited vertices of S, we can partition $\mathcal{C}^G$ into 3 subsets $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$. Furthermore, the set $\mathcal{C}_3$ can be partitioned into two subsets, $\mathcal{C}_3^{in}$ and $\mathcal{C}_3^{out}$, determined by whether the cycles are fully contained in S or not. Since (y, x) belongs to L_C^G, there is a convex combination of cycles of $\mathcal{C}^G$ whose coefficients satisfy

$$\sum_{\tau \in \mathcal{C}_1} \lambda_\tau^1 + \sum_{\tau \in \mathcal{C}_2} \lambda_\tau^2 + \sum_{\tau \in \mathcal{C}_3^{out}} \lambda_\tau^{3o} + \sum_{\tau \in \mathcal{C}_3^{in}} \lambda_\tau^{3i} + \lambda_{(0,0)} = 1 \tag{3.28}$$

Since the cycles in $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3^{out}$ have edges in $(t : S)$ and $(w : S)$, by the Lemma 3.1, each cycle has exactly one edge in the mentioned edge sets. Now, consider the hypothesis that $x(t : S) = c$ (or $x(w : S) = c$), so the coefficients also satisfy the following identity:

$$\sum_{\tau \in \mathcal{C}_1} \lambda_\tau^1 + \sum_{\tau \in \mathcal{C}_2} \lambda_\tau^2 + \sum_{\tau \in \mathcal{C}_3^{out}} \lambda_\tau^{3o} = c \tag{3.29}$$

By hypothesis, we have that $\lambda_{(0,0)} = 1 - c$ and by (3.28) and (3.29), we obtain that $\lambda_\tau^{3i} = 0$ for all $\tau \in \mathcal{C}_3^{in}$, which means that every cycle in $\mathcal{C}^G$ contained in S has null coefficient. □

Theorem 3.5 (Rules C1, C2 and C3)**.** *Given a vector $(y, x) \notin P_C^G$, let $S \subset V$ with $2 \leq |S| \leq 3$, $t \in V - S$ and $0 < c \leq 1$ be such that:*

(i) $y_v = c \ \forall v \in S \cup \{t\}$

(ii) $x(E(S)) = c \cdot (|S| - 1)$

(iii) $x(t : S) = c$

Then it is safe to shrink S for (y, x).

Proof. Let $W = V - (S \cup \{t\})$ be a subset of V. If the hypotheses are satisfied, note that W is non-empty. Since $2 \leq |S| \leq 3$, we have that $4 \leq |V[W]| \leq 5$. Notice that, $y_v = c$ for all the vertices of $V[W]$ and $(y[W], x[W]) \in P_A^{G[W]}$. Under these hypotheses, by Lemma 3.3, the vector $(y[W], x[W])$ belongs to $L_C^{G[W]}$. When $|S| = 2$, it does not exist any cycle contained in S. When $|S| = 3$, as a consequence of Lemma 3.4, we have that it does not exist a cycle in the convex combination of $(y[W], x[W])$ contained in S. Therefore, the hypotheses of Lemma 3.2 are satisfied and S is shrinkable. □

Shrinking rules for P_C^G

From Theorem 3.5, three shrinking rules can be derived, which are summarized in Figure 3.6: the rules C1 and C2 correspond to the case $|S| = 2$ and the rule C3 to $|S| = 3$.

Figure 3.7 shows the resulting graph after applying the C1 shrinking strategy to the support graph in Figure 3.2, while Figure 3.8 shows its topological representation.

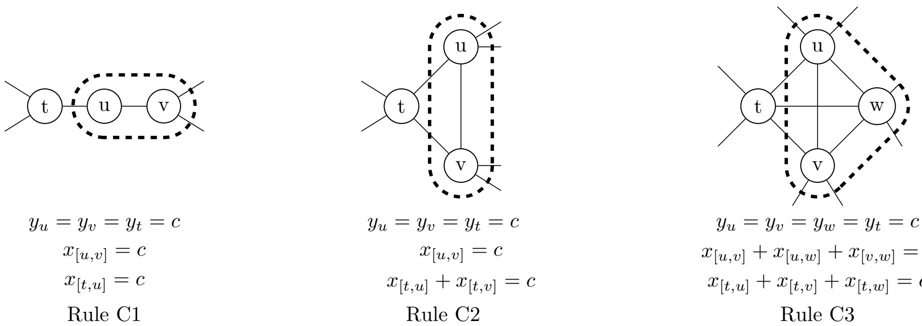


Figure 3.6: Illustration of the three shrinking rules derived from the Theorem 3.5

It is easy to see that rule C2 dominates the rule C1, in fact it is just a particular case of it. The reason to split them, is that the cost of checking C1 is lower than the cost of C2. By contrast, rule C3 is not dominated by the rules C1 and C2. In Figure 3.9, an example is given of a vector $(y, x) \in P_A$ in which rule C3 can be applied but not C1 and C2. For instance, if we consider $S = \{1, 2, 3\}$, $W = \{4, 5, 6\}$ and $t = 7$, then S is shrinkable by rule C3. Since the vertices and edges have different values, there is no shrinkable set that can be identified by rule C1 or C2.

A useful property of the rules derived from Theorem 3.5 is that the value of the vertices is inherited in the shrunk graphs.

Lemma 3.6. *Under the hypotheses of Theorem 3.5, $y[S](v[S]) = y_v$ for all $v \in V$.*

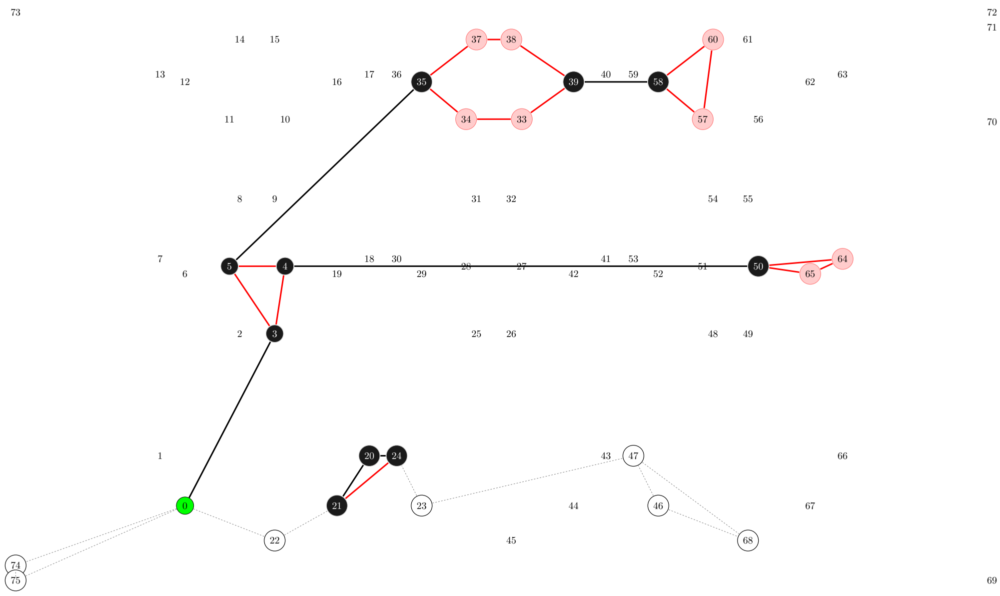


Figure 3.7: Resulting graph after C1 shrinking strategy

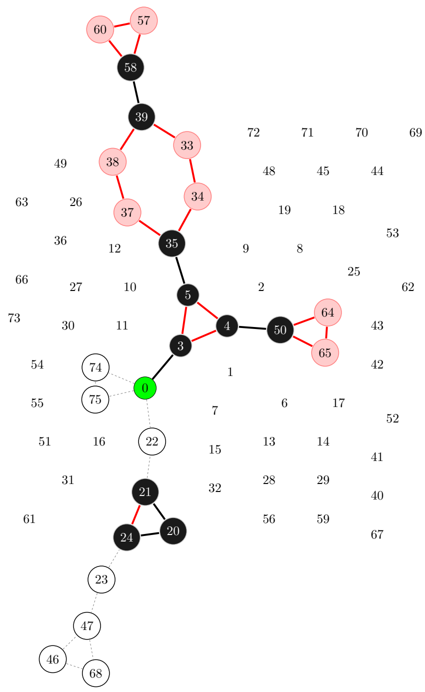


Figure 3.8: Topological representation of the graph after C1 shrinking strategy

Proof. For every $v \in V - S$, we have $y[S](v[S]) = y_v$ by definition. Since $2y_s = x(\delta(S)) = 2y_v$ for $v \in S$ we obtain the result of the lemma. $\square$

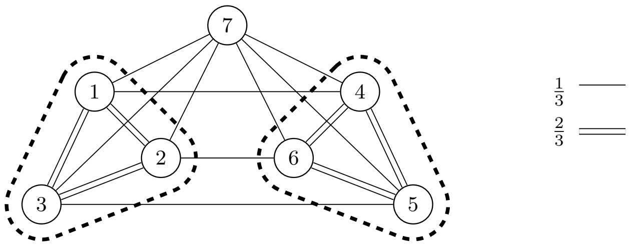


Figure 3.9: Example of a pair G and $(y, x) \in P_A$ where rule C3 can be applied but not rules C1 nor C2. The values of the edges are the ones detailed in the legend and all the vertices have value 1.

In the preprocess of separation algorithms, it is desirable to perform multiple consecutive safe shrinkings. For that aim, we need to analyse what happens with the hypotheses of Theorem 3.5 after the contraction of a shrinkable set. More precisely, we need to see when the shrunk vector belongs to P_A^G.

Lemma 3.7. *Let S be a shrinkable set for $(y, x) \in P_A^G$ obtained from Theorem 3.5 using the $\{S, W, \{t\}\}$ partition. Then, $(y[S], x[S])$ satisfies the degree equations and the logical constraints associated with every edge in $E(W) \cup (t : V)$. In addition, we have either*

i) $(y[S], x[S]) \in P_A^{G[S]}$, *or*

ii) $\exists w \in W$ *such that* $y_w < y_s$ *and* $y_w < x_{[w,s]} \leq y_s$

Proof. From the definition of the shrunk vector, it is clear that $(y[S], x[S])$ satisfies the degree equations. Since $v \in S$ satisfies $y_v \leq 1$, $y_s = y_v$ also satisfies $y_s \leq 1$. Moreover, $x_{[t,s]} = y_s = y_t$. If $x(w : S) \leq y_w$ for all $w \in W$ then $(y[S], x[S])$ satisfies the logical constraints and $(y[S], x[S]) \in P_A^{G[S]}$. If the previous is not true, there exists a vertex $w \in W$ such that $x(w : S) > y_w$ and $y_w < y_s$ (because by hypothesis $(y, x) \in P_A^G$). Therefore, the logical constraint $x_{[w,s]} \leq y_w$ is violated for $(y[S], x[S])$ by a vertex $w \in W$ such that $y_w < y_s$. □

There are two scenarios where the shrunk vector always belongs to P_A^G. First, when all the vertices of V have the same y value, as is the case when $(y, x) \in P_{TSP}^G$, and secondly, when only rule C1 is applied. The next theorem shows that if $(y, x) \in P_A^G$, it is possible to shrink a subset S obtained by the rules of Theorem 3.5 and continue with further safe shrinkings regardless of whether or not $(y[S], x[S])$ belongs to $P_A^{G[S]}$.

Theorem 3.8. *Given a vector $(y, x) \in P_A^G$, it is safe to consecutively apply the shrinking rules derived from Theorem 3.5.*

Proof. Let S be a subset obtained from Theorem 3.5 such that $(y[S], x[S]) \notin P_A^{G[S]}$. By Lemma 3.7 we know that the only violated logical constraints of $(y[S], x[S])$ consist of edges whose vertices, s and $v \in W$, have different values $y_v < y_s$. Notice that in the proof of Theorem 3.5 the hypothesis that the logical constraints are satisfied is used twice. First in Lemma 3.1, which is applied for vertices having the same value. Secondly in Theorem 3.5, where it is assumed $(y[N], x[N]) \in P_A^{G[N]}$ for a given subset N of $V[S]$. In order to see that this last hypothesis is always satisfied by every shrinkable set candidate, let us suppose that $\{M, N, \{r\}\}$ is a partition of $V[S]$ that satisfies hypotheses i), ii) and iii) of Theorem 3.5. Then there are two possible cases: $v \in M \cup \{r\}$ and $s \in N$, or vice versa. The hypothesis $(y[N], x[N]) \in P_A^{G[N]}$ is satisfied in both cases, because $x_{[n,v]} \leq y_n = y_u$ for $u \in M \cup \{r\}$. □

Another interesting scenario occurs when there is at least a vertex $v \in V$ satisfying $y_v = 1$, as happens in the context of cycle problems with depot. In all these problems, the case ii) of Lemma 3.7 has a special meaning as shown in Theorem 3.10.

Lemma 3.9. *If $(y, x) \in \mathbb{R}^{V \times E}$ satisfies the degree equations (3.7a) and $u, v \in V$ are two vertices such that $x_{[u,v]} > y_u$ then $x(\delta(\{u, v\})) < 2y_v$.*

Proof. As (y, x) satisfies the degree equations:

$$2y_u < 2x_{[u,v]} = 2y_u + 2y_v - x(\delta(\{u, v\})) \tag{3.30}$$

□

Theorem 3.10. *Given a vector $(y, x) \in P_A^G$, let $O = \{v \in V : y_v = 1\}$ be the subset of vertices with value equal to one and S be a shrinkable set for (y, x) obtained from Theorem 3.5 such that $O - S \neq \emptyset$. Then, we have either*

- *i) $(y[S], x[S]) \in P_A^{G[S]}$, or*
- *ii) $\exists w \in V - S$ such that, for every $u \in S$ and $v \in O - S$, the SEC $\langle S \cup \{w\}, u, v\rangle$ is violated by (y, x).*

Proof. Note that, in the case ii) of Lemma 3.7, the vertex $w \in V - S$ cannot be contained in O because $y_w < 1$. Now, as a consequence of Lemma 3.9 we can rewrite the second case. □

3.3.2 Safe Shrinking Rules for the Subcycle Closure Polytope

Depending on the inequality, more aggressive contractions can be employed as a preprocess of separation algorithms. In the TSP, for the subtour separation problem, [Crowder and Padberg, 1980] introduced subtour specific shrinking rules to simplify the support graphs before proceeding with the separation algorithms. With the aim of motivating the concepts in the subcycle-safe shrinking procedure, let us prove the following result.

Lemma 3.11. *Given a vector $(y, x) \in P_A^G$ and an edge $e \in E$, let $S = V(e)$ be the subset associated with the edge e. If $(y[S], x[S]) \in P_{SEC}^{G[S]}$, then either* *i)* $(y,x) \in P^G_{SEC}$, *or*

ii) every violated SEC $\langle Q,r,t\rangle$ *for* (y,x) *satisfies* $S \cap Q \neq \emptyset$ *and* $S - Q \neq \emptyset$

Proof. Let $e = [u,v]$ be the given edge and $\langle Q,r,t\rangle$ be a SEC for (y,x) such that $S \subset Q$ (or $S \subset V - Q$). On the one hand, since $(y,x) \in P^G_A$, we have $y[S](u[S]) \geq y_u$ and $y[S](v[S]) \geq y_v$. On the other hand, $x[S](\delta(Q[S])) = x(\delta(Q))$ by definition. Then the SEC $\langle Q[S], r[S], t[S]\rangle$ for $(y[S], x[S])$, is at least as violated as $\langle Q,r,t\rangle$ for (y,x). So if $(y[S], x[S]) \in P^{G[S]}_{SEC}$, and $(y,x) \notin P^G_{SEC}$, the only violated SECs for (y,x) are associated with subsets that separate u and v. □

Recall that we want to search the violated SECs for a vector $(y,x) \in P^G_A$, which has been obtained from the LP_0 subproblem. Let us assume that we have defined a first shrinking rule that contracts edges by avoiding the scenario ii) of Lemma 3.11. So if $(y,x) \notin P^G_{SEC}$, as a consequence of the lemma, $(y,x) \notin P^{G[S]}_{SEC}$. In this case, the vector $(y[S], x[S])$ does not belong to the closure of SECs because either there exists violated logical constraints, SECs or both. Let us suppose that we have a second shrinking rule that identifies (and saves) the violated logicals and "fixes" them. Repeatedly applying the second rule, we will eventually reach a vector that satisfies the logical constraints. Now, we are in a similar situation to the starting point, so we can try with the first rule again and so on. This is the main idea exploited in the subcycle-safe shrinking process.

Definition 3.2. *Given a vector* $(y,x) \in \mathbb{R}^{V \times E}$ *that satisfies the degree equations, a set* $S = \{u,v\} \subset V$ *is subcycle-safe to shrink if at least one of the following conditions is satisfied:*

i) $(y[S], x[S]) \notin P^{G[S]}_{SEC}$, *or*

ii) if there exist violated logical constraints for (y,x), *these are associated with the edge* $[u,v]$

Note that the second condition does not require the existence of violated logical constraints for (y,x), which enables the subcycle-safe shrinkable set definition for vectors (y,x) in P^G_{SEC} to be used. Furthermore, this condition means: if we have already found a violated constraint, we should not worry if later the shrinking the vector is projected to the subcycle closure polytope, since we have already achieved the goal of the separation problem.

In some sense, from Theorem 3.13 we derive the first shrinking rule of the motivation above and from Theorem 3.14 the second shrinking rule. The condition that avoids the case ii) of the Lemma 3.12 is the hypothesis $x_{[u,v]} \geq \max\{y_u, y_v\}$ in the theorems. Actually, the hypothesis that $(y,x) \in P^G_A$ of the first rule can be replaced with the hypothesis that all the logical constraints associated with vertices u and v (excluding the one with $[u,v]$) are satisfied, which is a consequence of the hypothesis $x_{[u,v]} \geq \max\{y_u, y_v\}$. Let us address the next lemma as an intermediate step.

Lemma 3.12. *Given a vector* $(y,x) \in \mathbb{R}^{V\times E}$ *that satisfies the degree equations, let* $S = \{u,v\} \subset V$ *be a subset such that* $x_{[u,v]} \geq \max\{y_u, y_v\}$*. Then, if* $(y,x) \notin P_A^G$*, at least one of the following conditions is satisfied:*

i) $(y[S], x[S]) \notin P_A^{G[S]}$*, or*

ii) if there exist violated logical constraints for (y,x)*, these are associated with the edge* $[u,v]$

Proof. On the one hand, since $x(\{u,v\} : w) \geq x_{[u,w]}$ and $x(\{u,v\} : w) \geq x_{[v,w]}$ for all $w \in V - \{u,v\}$, every violated logical constraint for (y,x) associated with the vertices in $V - \{u,v\}$ can be adapted to violated constraints for $(y[S], x[S])$. On the other hand, since $x_{[u,v]} \geq \max\{y_u, y_v\}$ and the degree equations are satisfied, we have that $x_{[u,w]} \leq y_u$ and $x_{[v,w]} \leq y_v$ for all $w \subset V - \{u,v\}$. Therefore, if $(y[S], x[S]) \in P_A^{G[S]}$, the only possible violated logical constraints associated with the vertices of S correspond with the edge $[u,v]$. □

The SEC inequalities (3.7c) are defined for sets, Q, such that $3 \leq |Q| \leq |V| - 3$. However, if $\langle Q,u,v\rangle$ violates for (y,x) the inequality of (3.7c) but $|Q| = 2$ or $|Q| = |V|-2$, then a violated logical constraint can be identified and therefore we also know that $(y,x) \notin P_{SEC}^G$. For instance, if $\langle\{u,w\},u,v\rangle$ does not satisfy the inequality (3.7c), then $x_{uw} \leq y_w$ is a violated constraint. In the following proofs, the term violated SEC, embracing the cases $|Q[S]| = 2$ and $|Q[S]| = |V[S]| - 2$, refers to its associated violated logical constraint when required.

Theorem 3.13 (Rule S1)**.** *Given a vector* $(y,x) \in \mathbb{R}^{V\times E}$ *that satisfies the degree equations, let* $u,v \in V$ *be two vertices such that* $x_{[u,v]} = y_u = y_v = c$*. If there exists a vertex* $w \in V - \{u,v\}$ *such that* $y_w \geq c$*, then it is subcycle-safe to shrink* $S = \{u,v\}$*.*

Proof. Assume the vector (y,x) belongs to P_A^G, i.e., only violated SECs exists for (y,x), otherwise the theorem is satisfied by Lemma 3.12. Let $\langle Q,r,t\rangle$ be a violated SEC for (y,x), and without loss of generality, suppose that $S \cap Q \neq \emptyset$. The goal is to see that for a violated SEC for (y,x), there is a violated SEC for $(y[S], x[S])$.

First, let us suppose that $S \subset Q$, where $x[S](\delta(Q[S])) = x(\delta(Q))$ is satisfied by definition. The only case that is needed to check is when $r \in S$. Without loss of generality, suppose that $r = v$. By hypothesis $y_u = x_{[u,v]}$, so $2y_v = x(\delta(S)) = 2y[S](v)$ and $\langle Q[S], y[S](s), y[S](r)\rangle$ define the desired SEC for $(y[S], x[S])$.

$$x[S](\delta(Q[S])) = x(\delta(Q)) < 2y_v + 2y_t - 2 = 2y[S](s) + 2y[S](t) - 2 \tag{3.31}$$

Next, let us analyze the case $S \cap Q \neq \emptyset$ and $Q - S \neq \emptyset$. Without loss of generality, suppose that $u \in Q$ and $v,w \in V - Q$. The subcase that requires a special attention is when $r = u$ and $t = v$. Note that, since (y,x) satisfies the degree equations and, also by hypothesis, $y_v = x_{[u,v]}$, we have that $x(v : V - Q) \leq x(v : Q)$, and therefore:

$$x[S](\delta(Q[S])) = x(\delta(Q \cup S)) \tag{3.32a}$$

$$= x(\delta(Q)) + x(\delta(v)) - 2x(v : Q) \tag{3.32b}$$

$$= x(\delta(Q)) + x(v : V - Q) - x(v : Q) \leq x(\delta(Q)) \tag{3.32c}$$

$$< 2y_r + 2y_v - 2 = 2y_w + 2y_t - 2 = 2y[S](r) + 2y[S](w) - 2 \tag{3.32d}$$

Hence, there also exists a violated SEC (or logical constraint) for $(y[S], x[S])$ and the set S is subcycle-safe to shrink. □

Figure 3.10 shows the resulting graph after applying the S1 shrinking strategy to the support graph in Figure 3.2, while Figure 3.11 shows its topological representation.

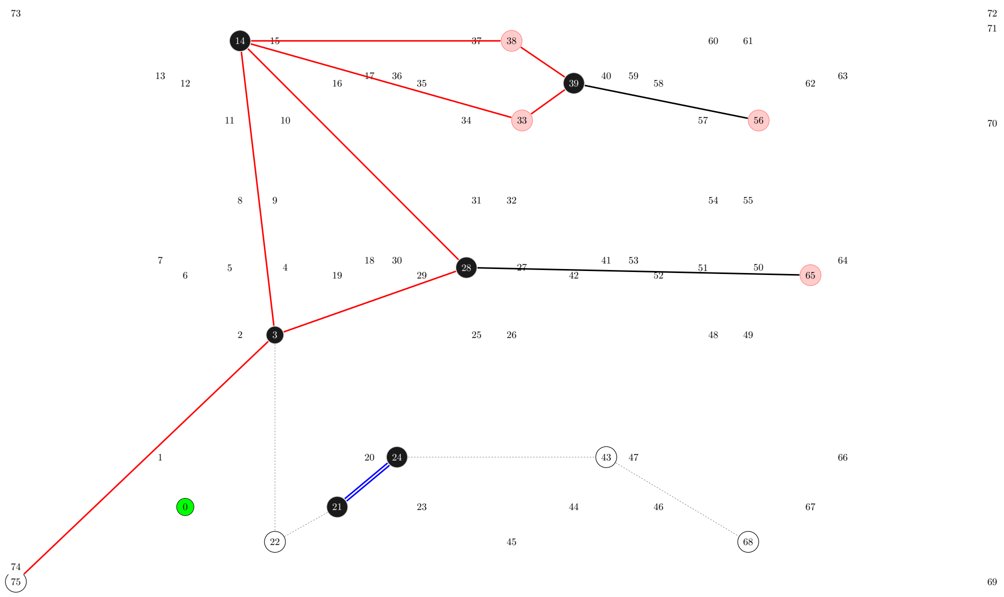


Figure 3.10: Resulting graph after S1 shrinking strategy

Clearly, the shrinking rule S1 dominates the rules C1 and C2 of Theorem 3.5. For every scenario where rules C1 or C2 can be applied, rule S1 is also applicable, since the existence of $w \in V - \{u, v\}$ is determined by the vertex $t \in V - \{u, v\}$ in Theorem 3.5. Moreover, rule C3 should not be combined with rule S1, since might exist vertices with the same y value whose connecting edge has a greater value in the shrunk graph obtained by S1.

Theorem 3.14 (Rule S2)**.** *Given a vector $(y, x) \in \mathbb{R}^{V \times E}$ that satisfies the degree equations, let $u, v \in V$ be two vertices such that $x_{[u,v]} > \max\{y_u, y_v\}$ then it is subcycle-safe to shrink $S = \{u, v\}$.*

Proof. The theorem is a direct consequence of Lemma 3.12. □

Note that, if $(y, x) \in P_A^G$ and S is a shrinkable set obtained from Theorem 3.5, then by Lemma 3.7 we have that $x_e \leq \max\{y_u, y_v\}$ for every $e = [u, v] \in E[S]$. Hence, it only makes sense to use the rule S2 in combination with the rule S1.

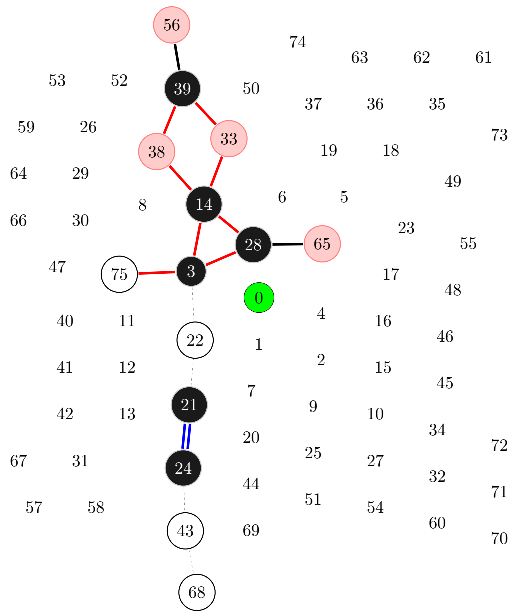


Figure 3.11: Topological representation of the graph after S1 shrinking strategy

Figure 3.12 shows the resulting graph after applying the S1 shrinking strategy to the support graph in Figure 3.2, while Figure 3.13 shows its topological representation.

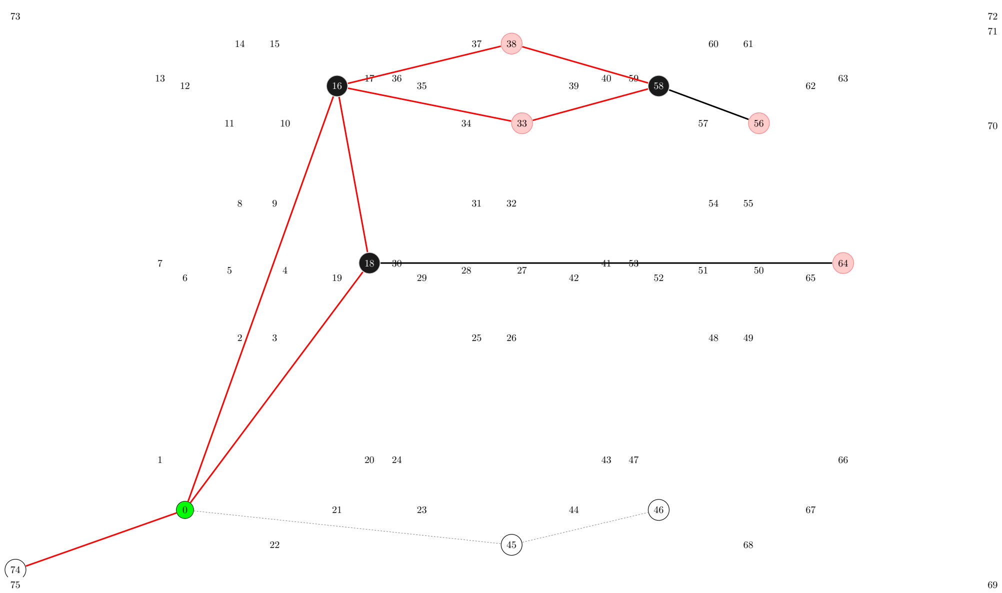


Figure 3.12: Resulting graph after S1S2 shrinking strategy

If a subcycle-safe rule is applied, we know that all the SECs have not vanished. However, new violated SECs for $(y[S], x[S])$ might have appeared, which cannot be adapted

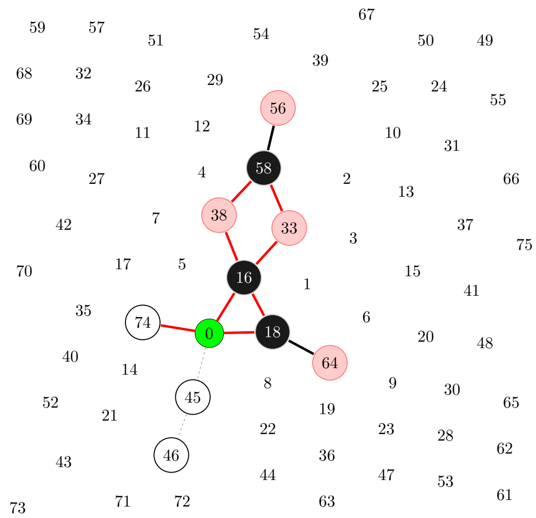


Figure 3.13: Topological representation of the graph after S1S2 shrinking strategy

to a violated one for (y, x). This situation would lead to identifying unnecessary cuts for (y, x) and therefore to slowing down the separation algorithm (the cut generation part). It is reasonable to ask when the violated SECs for $(y[S], x[S])$ can be transformed to violated SECs for (y, x) and when not. Let us define the mapping by $\pi_S : \mathcal{P}(V[S]) \to \mathcal{P}(V)$

$$\pi_S(Q) = \begin{cases} Q - \{s\} \cup S & \text{if } s \in Q \\ Q & \text{otherwise} \end{cases} \tag{3.33}$$

For a given S, the inverse, π_S^{-1}, of the mapping π_S is the set shrinking defined in (3.18), i.e., $\pi_S^{-1}(Q) = Q[S]$. We have that $Q = \pi_S^{-1}(\pi_S(Q))$ for all $Q \subset V[S]$ and $Q \subset \pi_S(\pi_S^{-1}(Q))$ for all $Q \subset V$. An important property of the mapping π_S, by the definition (3.19c), is that $x(\delta(\pi_S(Q))) = x[S](\delta(Q))$ for all $Q \subset V[S]$. In some cases, we will need to refer to the set obtained by unshrinking completely the contracted sets, where multiple shrinking might have been performed, e.g., $G[S_1][S_2]$. In such cases, we simplify the notation and denote $\pi(Q)$, e.g., $\pi(Q) = \pi_{S_1}(\pi_{S_2}(Q))$.

When an inequality family is targeted in a separation problem, knowing the representation of such inequalities, as is the case for the SECs, is very valuable to study how an inequality is transformed when shrinking and unshrinking a set. Moreover, since $x(\delta(\pi_S(Q))) = x[S](\delta(Q))$ for all $Q \subset V[S]$, understanding the relationship between y and $y[S]$ values is the key point to see how the violated SEC inequalities behave under the different shrinking rules.

Lemma 3.15. *Given a vector $(y, x) \in \mathbb{R}^{V \times E}$ that satisfies the degree equations and a subset $S = \{u, v\}$ of V. The following holds:*

i) $y[S](v[S]) > y_v$ *if* $x_{[u,v]} < y_u$

ii) $y[S](v[S]) < y_v$ *if* $x_{[u,v]} > y_u$

iii) $y[S](v[S]) = y_v$ *if* $x_{[u,v]} = y_u$

Proof. It is a consequence of the definition of $y[S]$ and the identity (3.14). □

Lemma 3.16. *Under the hypotheses of Theorem 3.13,* $y[S](v[S]) = y_v$ *for all* $v \in V$*.*

Proof. For every $v \in V - S$, we have $y[S](v[S]) = y_v$ by definition. For $u, v \in S$, since $y_u = y_v = x_{[u,v]}$, we obtain the equality by Lemma 3.15. □

Lemma 3.17. *Let* G *be an undirected graph,* $(y, x) \in \mathbb{R}^{V \times E}$ *be a vector and a vertex subset* $S \subset V$*. Suppose that* $y[S](u) \leq y(v)$ *for all* $u \in V[S]$ *and* $v \in \pi_S(u)$*. Then, for each SEC for* $(y[S], x[S])$ *there exists at least one SEC as violated as it for* (y, x)*.*

Proof. Note that, if $r \in Q$ and $t \notin Q$ then $u \in \pi_S(Q)$ and $v \notin \pi_S(Q)$ for all $u \in \pi_S(r)$ and $v \in \pi_S(t)$. Let $\langle Q, r, t\rangle$ be a SEC inequality violated by $(y[S], x[S])$. Therefore, the SEC inequality $\langle \pi_S(Q), u, v\rangle$ is violated by (y, x) where $u \in \pi_S(r)$ and $v \in \pi_S(t)$.

$$x(\delta(\pi_S(Q))) - 2y_u - 2y_v \leq x[S](\delta(Q)) - 2y[S](r) - 2y[S](t) \qquad u \in \pi_S(r) \text{ and } v \in \pi_S(t) \tag{3.34}$$

□

Corollary 3.18. *Let* G *be an undirected graph and* $(y, x) \in \mathbb{R}^{V \times E}$ *be a vector. If* S *is a shrinkable subset obtained by rules C1, C2, C3 or S1, then* $(y, x) \notin P^G_{SEC}$ *if and only if* $(y[S], x[S]) \notin P^{G[S]}_{SEC}$*.*

Proof. It is a consequence of Lemma 3.6 and Lemma 3.16. □

When rule S2 is applied, as a consequence of Lemma 3.15, some vertices of the shrunk graph will have lower values than the original ones. Although, by the definition of subcycle-safe shrinking, all the violated SECs for (y, x) are not vanished, we might lose some of them in the shrinking process. However, it could be interesting to identify and save those excluded violated SECs if possible. For that aim we consider a vector $m[S] \in \mathbb{R}^{V[S]}$ defined as $m[S](v) = \max\{y_u : u \in \pi_S(v)\}$. It is clear that if only the rules of Theorem 3.5 and the rule S1 are applied, $m[S](v) = y[S](v)$ for all $v \in V[S]$. Considering the vector $m[S]$, we evaluate a SEC $\langle Q, u, v\rangle$ for a given vector $(y[S], x[S])$ by the expression

$$x[S](\delta(Q)) - 2m[S](u) - 2m[S](v) \geq -2 \tag{3.35}$$

and only if this is violated, we save the SEC $\langle Q, u, v\rangle$ for (y, x).

3.4 Separation Algorithms for Subcycle Elimination Constraints

In this section, we present two exact separation algorithms for SECs in cycle problems. Given a vector $(y, x) \in P_A^G$, an algorithm which finds violated SECs for (x, y) is called a separation algorithm for SECs. A separation algorithm is called exact if it always finds violated inequalities when they exist, otherwise it is called heuristic.

Before delving into the separation algorithms in depth, we need to make an observation which has important consequences for SEC separation problems in cycle problems. In the TSP, the y values are fixed to 1, so the constraints in the family (3.7c) only depend on the star-set value of subsets of vertices. For this reason, the SEC separation problem for the TSP is closely related with the minimum cut problem, particularly, the most violated SEC for (y, x) is in correspondence with the global minimum cut of G^*.

SEC separation problem and minimum cut problem

In cycle problems in general, the SECs $\langle C, v, d \rangle$ obtained from the global minimum cut of G^*, $x(C : V - C)$, might not be violated, although other violated SECs for (y, x) can exist.

This scenario is shown in the example in Figure 3.14. The global minimum cut in the figure is obtained by $C = \{4\}$ and because $|C| < 3$, by definition (3.7c), there is no violated SEC inequality of type $\langle C, v, u \rangle$ (or equivalently of type $\langle V - C, v, u \rangle$). However, the SECs $\langle \{2, 3, 8\}, 2, 6 \rangle$ (or $\langle \{1, 4, 5, 6, 7, 9\}, 6, 2 \rangle$), $\langle \{2, 3, 4, 8\}, 2, 6 \rangle$ (or $\langle \{1, 5, 6, 7, 9\}, 6, 2 \rangle$) and $\langle \{2, 3, 4, 5, 8\}, 2, 6 \rangle$ (or $\langle \{1, 6, 7, 9\}, 6, 2 \rangle$) are violated for the vector (y, x) represented in Figure 3.14.

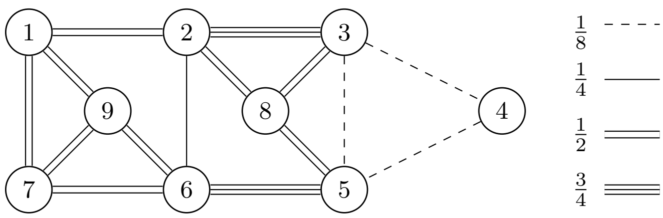


Figure 3.14: An example of a vector (y, x) where the associated SEC with the global minimum cut of the support graph is not violated, while violated SECs for the vector exist. The edge values of the vector (y, x) are detailed in the legend, while the vertex values are derived by the degree equations.

The straightforward exact algorithm to find violated SECs for (y, x), consists of solving $\binom{|V^*|}{2}$ number of (s, t)-minimum cuts problems on G^*, one for each pair of different vertices, and then evaluating the associated inequality (3.7c) using the y values of the pair of vertices. When using the push-relabel algorithm in [Goldberg and Tarjan, 1988] with highest-level selection and global relabeling heuristics to solve the (s, t)-minimum cut problems (or better said, to solve its dual: the (s, t)-maximum flow problems), the straightforward exact strategy has a $O(|V^*|^4\sqrt{|E^*|})$ time complexity. Note that for cycle problems in general, the algorithm in [Hao and Orlin, 1992] cannot be used to find the most violated SEC. Although this algorithm solves the global minimum cut in $O(|V^*|^2\sqrt{|E^*|})$ steps, which might be very useful, particularly for the TSP, in a general cycle problem the global minimum cut might not correspond with a violated SEC as shown above.

The proposed separation algorithms in this chapter, the Dynamic Hong's algorithm and the Extended Padberg-Grötschel algorithm, are two exact algorithms for cycle problems that run in $O(|V^*|^3\sqrt{|E^*|})$. They are motivated by two observations made in [Fischetti et al., 1997]. First, for a given pair of different vertices $u, v \in V$, the most violated SEC, $\langle Q, u, v\rangle$, corresponds to the subset Q such that $(Q : V - Q)$ is a (u, v)-minimum cut. Secondly, for a given subset Q, the most violated SEC, $\langle Q, u, v\rangle$, corresponds to the vertices $u = \arg\max\{y_w : w \in Q\}$ and $v = \arg\max\{y_w : w \in V - Q\}$. The next two algorithms exploit these two observations, in order to guarantee that the most violated SEC for (y, x) is identified.

3.4.1 Dynamic Hong's Exact Separation Algorithm

The Hong's exact approach, which emerged in the context of the TSP, consists of solving only $|V^*| - 1$ number of (s, t)-minimum cut problems, by fixing a random vertex, s, as the source of all the minimum cut problems, at the expense of possibly losing a subset of violated cuts, see [Hong, 1972].

This exact approach can be extended for cycle problems, by selecting s as a vertex of V^* with maximum y value. Based on the second observation in [Fischetti et al., 1997], an s selected this way will belong to the most violated SEC corresponding to every subset Q. However, since to define a SEC we need to select another vertex in $V^* - \{s\}$, based on the first observation, we consider for each $t \in V^* - \{s\}$ the subset Q such that $(Q : V - Q)$ is a (s, t)-minimum cut. This shows that the extension of the Hong's approach for cycle problems is also an exact separation algorithm.

Let us suppose that the vertices $V^* = \{v_1^*, \ldots, v_{|V^*|}^*\}$ are ordered decreasingly by y and define the source $s_i = v_1^*$ and the sink $t_i = v_{i+1}^*$ for all $i \in \{1, \ldots, |V^*| - 1\}$. In [Fischetti et al., 1998] and [Bérubé et al., 2009], after each (s_i,t_i)-minimum cut, $(Q : V - Q)$, they increase the weight of the edge $[s_i, t_i]$ by $2 - x(\delta(Q))$, in order to prevent collecting the same SEC in subsequent iterations. A disadvantage of this strategy is that the degree equations are not satisfied anymore. In Theorem 3.20 we achieve the same objective by shrinking the set $\{s_i, t_i\}$, with the extra feature of reducing the size of the graph for the following iterations.

The underlying idea of Theorem 3.20 comes from the shrinking rule for minimum cut problems, Theorem 3.3, in [Padberg and Rinaldi, 1990a]. This theorem says that the edges having a value greater than or equal to the upper bound of the minimum cut can be contracted. However, this rule is not safe for SECs in cycle problems. For instance, based on Theorem 3.3, in Figure 3.14 we would shrink the set $\{2, 6\}$ because the value of the edge $[2, 6]$ is equal to the global minimum cut value $x(C : V - C)$. However, because all the violated SECs in the figure consider the vertices 2 and 6 as disjoint ones, it is not safe to shrink the set $\{2, 6\}$.

Lemma 3.19. *Given a vector $(y, x) \in \mathbb{R}^{V \times E}$ that satisfies the degree constraints and four vertices $u, v, u', v' \in V^*$ such that $y_u + y_v \geq y_{u'} + y_{v'}$, let $(Q : V^* - Q)$ be a (u, v)-minimum cut and $(Q' : V^* - Q')$ be a (u', v')-minimum cut in G^*. If $\langle Q', u', v' \rangle$ is a strictly more violated SEC than $\langle Q, u, v \rangle$, then both u, v vertices belong either to Q' or $V^* - Q'$.*

Proof. Suppose that $\langle Q', u', v' \rangle$ is a strictly more violated SEC than $\langle Q, u, v \rangle$, then:

$$x(\delta(Q)) - 2y_u - 2y_v + 2 > x(\delta(S)) - 2y_{u'} - 2y_{v'} + 2 \quad (3.36a)$$

$$x(\delta(Q)) > x(\delta(S)) + 2y_u + 2y_v - 2y_{u'} - 2y_{v'} \quad (3.36b)$$

$$x(\delta(Q)) > x(\delta(S)) \quad (3.36c)$$

Since $x(\delta(Q)) = x(Q : V^* - Q)$ is the value of the (u, v)-minimum cut and $x(\delta(Q'))$ is strictly smaller than it, then both u and v belong either to Q' or $V - Q'$. □

Theorem 3.20 (Rule S3)**.** *Given a vector $(y, x) \in \mathbb{R}^{V \times E}$ satisfying the degree equations, consider $u, v \in V^*$ such that $\min\{y_u, y_v\} \geq y_w$ for all $w \in V^* - \{u, v\}$. Then, after solving the (u, v)-minimum cut problem and collecting, if any, the associated violated SECs, it is subcycle-safe to shrink $S = \{u, v\}$.*

Proof. The theorem is a direct consequence of Lemma 3.19. □

The dynamic Hong's algorithm is based on Theorem 3.20, and it takes its name because the source, s, for the (s, t)-minimum cut problems might not be the same as in the classical approach. The algorithm works as follows: suppose that the vertices of V^* are ordered decreasingly by y, and set for the first minimum cut problem $s_1 = v_1^*$ and $t_1 = v_2^*$. Next, we solve the (s_1,t_1)-minimum cut problem, evaluate the obtained SEC candidates and, thereafter, shrink $\{s_1, t_1\}$. To proceed with the subsequent iteration, we need to know if the ordering of the vertices has changed after the $\{s_1, t_1\}$ shrinking, so we consider the Lemma 3.15. When the logical constraint $x_{[s_1,t_1]} \leq y_{s_1}$ is satisfied, we have that $y[\{s_1, t_1\}](s_1[\{s_1, t_1\}]) \geq y_{t_1} \geq y_v$ for all $v \in V^* - \{s_1, t_1\}$, and, hence, the vertex $s_1[\{s_1, t_1\}]$ will be "again" the source of the subsequent minimum cut problem. However, when $x_{[s_1,t_1]} > y_{s_1}$, it might happen that $y[\{s_1, t_1\}](s_1[\{s_1, t_1\}]) < y_v$ for some $v \in V^* - \{s_1, t_1\}$. In this situation, after shrinking the set $\{s_1, t_1\}$, we will need to reorder the vertices of $V^*[\{s_1, t_1\}]$ decreasingly by y (rearrange $s_1[\{s_1, t_1\}]$ in the set V^*). So now, to proceed, we set as s_2 and t_2, the first two vertices of $V^*[\{s_1, t_1\}]$, continue by solving the (s_2,t_2)-minimum cut problem, evaluating the possible violated SECs and shrinking $\{s_2, t_2\}$, and so on.

3.4.2 Extended Padberg-Grötschel Exact Separation Algorithm

[Padberg and Grötschel, 1985], showed a different exact separation algorithm for SECs in the TSP, whose key component is the multitermal flow algorithm proposed in [Gomory and Hu, 1961]. A multitermal flow algorithm is solved, in turn, using the so-called Gomory-Hu tree, which can be constructed solving a $|V^*|-1$ number of (s,t)-minimum cut problems.

In [Fischetti et al., 1997] it was mentioned that an analogue approach to the one given for the TSP might be used for the SECs in the cycle problems, but no details were given to illustrate how this approach should be extended. However, note that the adaptation of the Padberg-Grötschel approach for cycle problems is not trivial. The algorithm in [Padberg and Grötschel, 1985] for the TSP relies on the correspondence between the most violated subtour elimination constraint for (y,x) and the global minimum cut of G^*, which is not always the case in general cycle problems (this might not even be violated while other exist).

In cycle problems, Gomory-Hu trees were used to find violated SECs in [Bauer et al., 2002] for the Cardinality Constrained Cycle Problem (CCCP) and in [Jepsen et al., 2014] for the Capacitepd Profitable Tour Problem (CPTP). Nevertheless, in absence of details of the approach used to identify the violated SECs, we understand that in both papers the selected inequality corresponds with the global minimum cut. Therefore, these separation algorithms for SECs should be considered as heuristics. As far as we know, an exact extension for the Padberg-Grötschel separation algorithm for SECs in cycle problems has not been detailed in the literature.

In order to extend the separation algorithm for cycle problems, we need to construct a Gomory-Hu tree, $T=(V^*,A_T)$, of the support graph G^* with weights (y,x). However, unlike in the original approach, the tree T has to be constructed as a directed rooted tree, where the root is set as a vertex of V^* with maximum y value. Let us denote by $\Delta(v)$ the set of descendant vertices of $v \in V^*$ and by r the root of the tree T. We consider that every vertex is descendant of itself, i.e., $v \in \Delta(v)$. Suppose that the arcs of A_T are in the descendant orientation, and call h_e the head vertex of an arc a. Given $a \in A_T$, we define

$$u_a = \arg\max\{y_v : v \in \Delta(h_a)\} \tag{3.37a}$$

$$v_a = \arg\max\{y_v : v \in V^* - \Delta(h_a)\} \tag{3.37b}$$

which identifies the vertices, u_a and v_a, with the maximum y value for each of the two connected components of the graph $(V^*, A_T - \{a\})$. Note that, from the way that we have chosen the root, we can assume that $v_a = r$. Then, once the directed rooted Gomory-Hu tree is constructed, the violated SECs are collected in $O(V^*)$ computational time. With that aim, we check for each arc $a \in A_T$ ($|A_T| < |V^*|$) if the inequality $w_a - 2y_{u_a} - 2y_r \geq -2$ is violated, being w_a the weight of the arc a in the Gomory-Hu tree T representing the (s,t)-minimum cut for the two extreme vertices of the arc a. If this happens, the violated SEC is defined by $\langle \Delta(h_a), u_a, r\rangle$.

Note that this can be done efficiently because the u_a vertices of the arcs can be updated without an extra computational overhead. At every step of the Gomory-Hu algorithm, when a new arc is added to the tree, the descendant vertices are identified, which can be grasped to update the u_a vertices. Also, with a proper implementation of the Gomory-Hu algorithm, it is possible to maintain the subset that contains the selected r as the root of the subsequent trees. For more details, see the pseudocode in the Appendix A.1.3.

In a similar way to the extension of Hong's approach, it can be shown that the extension of Padberg-Grötschel is exact for cycle problems. In this case, the root vertex r plays the role of s, whereas each arc $a \in A_T$ identifies simultaneously a vertex in $V - \{r\}$, $t = h_a$, and its associated (s,t)-minimum cut. Furthermore, it goes one step beyond, based on the second observation, it considers u_a instead of h_a. Hence, the number of violated cuts found by the extension of the classical Hong's approach is dominated by the extension of the Padberg-Grötschel approach.

According to our experiments in Section 3.5, the Extended Padberg-Grötschel approach consumes a much lower computational time than the Extended Hong approach, although both approaches have the same worst case running time complexity. This happens because the subsequent (s,t)-minimum cut problems are solved in subgraphs of G^* in the Gomory-Hu tree based approach. When the problem size increases, the time needed for the shrinking and unshrinking operations during the Gomory-Hu tree construction is insignificant compared to the time needed to solve the (s,t)-minimum cut problems. Therefore, in addition to potentially finding more violated SECs, the Extended Padberg-Grötschel is a faster exact separation algorithm than the Extended Hong's Algorithm.

In Figure 3.15, we illustrate the Extended Padberg-Grötschel approach to find the violated SECs for the vector (y, x) defined in Figure 3.14. The weight w_a of each $a \in A_T$ in the tree is detailed above the arcs, and the y values of the vertices u_a and v_a are detailed inside a box, at the top and at the bottom respectively, near the head vertex of the arc. Two violated SECs are identified $\langle\{2,3,4,5,8\},2,6\rangle$ and $\langle\{2,3,8\},2,6\rangle$. Note that, if in this particular tree, the vertex 2 is chosen to be the root, only the violated SEC $\langle\{1,6,7,9\},6,2\rangle$ (equivalent to $\langle\{2,3,8\},2,6\rangle$) is collected, which shows that the exact algorithm is sensible to the directed rooted Gomory-Hu tree construction.

Although, the detailed approach until now always finds violated inequalities when they exist, extra violated SECs can be collected using a more exhaustive search whose cost is $O(|V^*|^2)$. Observe that $x(\delta(\Delta(h_a) \cup \Delta(h_f))) \leq w_a + w_f$ for every $a, f \in A_T$. Then, we can define $y_{u(e,f)} = \max\{y_{u_a}, y_{u_f}\}$ and check if $w_a + w_f - 2y_{u(a,f)} - 2y_r < -2$ for each pair arcs of A_T. This way, the violated SEC $\langle\{2,3,4,8\},2,6\rangle$ in Figure 3.15 can be identified. We have not made use of this kind of extra SECs in our experiments.

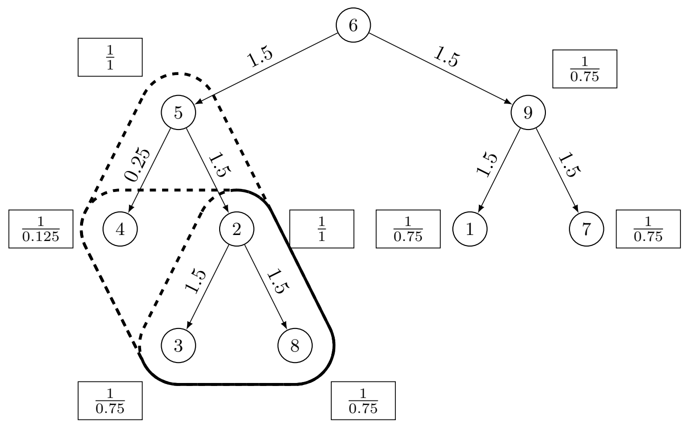


Figure 3.15: An example of the directed rooted Gomory-Hu tree for the SEC separation problem of Figure 3.14. The u_a (below) and v_a (above) values are detailed in the boxes. The arc weights are detailed next to the arcs.

3.5 Computational Experiments

In this section we describe the results of the computational experiments for the shrinking and the exact separation algorithms for SECs. These experiments have been designed with two goals in mind. First, to show the importance of the shrinking technique for cycle problems, and second, to evaluate the performance of different combination of shrinking and separation algorithms for SECs.

The computational study of this section is inspired by two studies for the minimum cut algorithms: [Jünger et al., 2000] and [Goldberg and Tsioutsiouliklis, 2001]. In both papers, the minimum cut algorithms are tested in instances originated, among others, from the solution of the TSP by a B&C algorithm. Note that, as explained in Section 3.4, the global minimum cut algorithms tested in these papers are not suitable for our aim.

[Jünger et al., 2000] studied the performance of different algorithms in combination with the shrinking rules defined for the minimum cut problems in [Padberg and Rinaldi, 1990a]. Similarly, in this chapter, we show the performance of the combination of shrinking rules and separation algorithms for SECs in cycle problems. [Goldberg and Tsioutsiouliklis, 2001] compared different Gomory-Hu tree building strategies: [Gusfield, 1990] implementation and three variants of the classical implementation. It was shown, for the SEC separation problem in the TSP, that the classical Gomory-Hu building based strategies outperform Gusfield's implementation, whereas they have not obtained significant differences among the variants of the classical implementation. The directed rooted Gomory-Hu tree algorithm presented in Section 3.4 can be considered within the class of classical implementations.

3.5.1 Benchmark Instances

The cycle problems could have a very large variety of origins, where the cycle constraints might be combined with additional constraints (e.g., a limit in the length of the cycle) and different objective functions (e.g., maximizing the profits and/or minimizing the length). These different natures of the cycle problems might vary the results obtained by each proposed strategy. However, we assume that in general terms the behaviour of the strategies for SECs is similar for all the cycle problems. So, instead of presenting an extensive comparison for different cycle problems, we focus our experiments on a well-known cycle problem, the Orienteering Problem (OP).

With the purpose of evaluating our shrinking and separation algorithms for SECs, we have built the SEC separation instances by obtaining vectors $(y, x) \notin P_C^G$ during a B&C algorithm for the OP. The OP instances are constructed based on the TSPLIB instances in [Reinelt, 1991] following the approach in [Fischetti et al., 1998]. Particularly, we have chosen the TSPLIB instances selected in [Goldberg and Tsioutsiouliklis, 2001]: pr76, att532, vm1084, rl1323, vm1748, rl5934, usa13509, d15112. Based on these 8 TSP instances, we have constructed 24 OP instances following the approach in the OP literature. The depot vertex is considered to be the first vertex of the TSPLIB instance, the maximum cycle length in the OP is set as half of the TSP value of the instance (values reported in [Applegate et al., 2007]) and the profits of the vertices are generated in three different ways: Gen1, all the vertices have equal profit; Gen2, the scores are generated pseudorandomly; and Gen3, the vertices which are further from the depot vertex have a greater profit. Once the OP instances have been constructed, the SEC separation instances are generated by considering the first support graph during a B&C algorithm for the OP which satisfies the degree constraints, the logical constraints and the connectivity. We have classified the instances into two equal-sized groups: Medium, instances whose original OP problem has less than 1500 vertices, and Large, the rest of the instances. All the used OP instances and SEC separation problem instances are available in `https://github.com/gkobeaga/cpsrksec`.

3.5.2 Shrinking Strategies for SECs

Relying on the results of Section 3.3.1 and Section 3.3.2, we have considered 5 different shrinking strategies for SECs. We have named the obtained strategies, by concatenating the names of the involved rules: C1, C1C2, C1C2C3, S1, S1S2. The pseudocodes of these strategies are detailed in Appendix A.

In each strategy, each involved rule is applied exhaustively. For instance, for the rule C1, the hypotheses of Theorem 3.5 are checked for every possible set $S \subset V^*$ and vertex $t \in V^* - S$. Moreover, when a shrinkable set S is found and shrunk, new shrinkable sets might appear in the graph obtained after applying the shrinking. In order to handle these scenarios, we make use of a heap set, $H \subset V^*$, which stores all the vertices that need to be checked to see whether they belong to a candidate S. For that, first, the set

H is initialized considering all the vertices of V^*. During the search procedure, whenever the heap set H is not empty, we draw one of its vertex, v, and consider it as contained in S. Then, we find neighbour vertices of v that, if they incorporate to S, might make S shrinkable. If a shrinkable set S is found, first we remove the vertices in the set S from H, and then we shrink the graph G^* and the vectors (y, x) and m (remember that $m_v = \max\{y_u : u \in \pi(v)\}$ for $v \in V^*$). Immediately thereafter, we add the newly created vertex s and its neighbours to the heap H. Additionally, when the support graph has vertices with value one, we check if violated SECs exist as suggested by Lemma 3.9 and Theorem 3.10.

3.5.3 Exact Separation Algorithms for SECs

We study the performance of four exact separation algorithms for SECs:

i) Algorithm EH: Extended Hong's algorithm.

ii) Algorithm DH: Dynamic Hong's algorithm.

iii) Algorithm DHI: Dynamic Hong's algorithm with internal shrinking.

iv) Algorithm EPG: Extended Padberg-Grötschel algorithm.

The Algorithm EH is the Hong separation algorithm extended for cycle problems in [Fischetti et al., 1997]. The Algorithm DH refers to the Dynamic Hong separation algorithm explained in Section 3.4, i.e., after each minimum cut, we shrink the source and sink vertices based on rule S3. In Algorithm DHI, in analogy to the approach used in [Applegate et al., 2007] for the TSP, inside the DH separation algorithm, after shrinking the source and the sink vertices, we apply the given shrinking strategy to the newly obtained graph. The Algorithm EPG refers to the extended Padberg-Grötschel algorithm explained in Section 3.4.

When a violated SEC, $\langle Q, u, v\rangle$, is found, we save in a repository only the Q set of the violated SEC. During the whole separation procedure each Q set is saved only once to avoid generating unnecessary cuts. Moreover, if $|Q| > |V^*|/2$, we save $V^* - Q$ instead of Q in order to decrease memory resource requirements. Once the separation algorithm is completed, we generate the SEC cuts from the saved Q sets in the following way: we consider for candidate vertices, u and v, the vertices with maximum y value inside Q, $M(Q) = \{u \in Q : y_u \geq y_v \ \forall v \in Q\}$, and outside Q, $M(V^* - Q) = \{u \in V^* - Q : y_u \geq y_v \ \forall v \in V^* - Q\}$. Since the amount of generated SECs might be huge (producing memory problems) and it is likely unnecessary to consider all of them, we consider only k_{in} and k_{out} randomly selected vertices from $M(Q)$ and $M(V^* - Q)$, respectively. Note that in a cycle problem with depot, we have either $d \in M(Q)$ or $d \in M(V^* - Q)$ for every Q, so it would be sufficient to select the depot instead of the randomly selected vertices. In other words, in these problems, it is enough to consider $u = d$ and $k_{in} = 1$ if $d \in M(Q)$ and $v = d$ and $k_{out} = 1$ otherwise. However, with the aim of obtaining insights about the SEC generation process in general cases, in the experiments, we have ignored that the OP is a cycle problem with depot.

The pseudocodes of the considered shrinking and separation strategies can be found in Appendix A and the source code of the implementation used for the experiments is publicly available in `https://github.com/gkobeaga/cpsrksec`.

3.5.4 Results

For the experiments, we have run 10 times each combination of shrinking and separation strategies with two objectives in mind: evaluate the influence of the random choices during the algorithm (ties are broken randomly when ordering V^*; source and sink vertices are selected randomly in the Gomory-Hu tree construction) and obtain a better approximation of the running times. We have divided the process of finding the violated cuts into three parts: (1) the preprocess, which considers the shrinking carried out before the separation, (2) the separation, which consists of finding the Q sets that define violated cuts, and (3) the generation of the violated SEC from the Q sets. Since the SEC generation is closely related to the obtained Q sets in the previous parts, and it is independent of the considered shrinking and separation strategies, we have limited the discussion of results to the preprocess and the separation parts.

The computational results are summarized in two tables. In Table 3.1, we present the information about the graph simplification and the relative time needed by each combination of strategies compared to the reference strategy (Algorithm EH with NO shrinking). In Table 3.2, we show the absolute values (on average) about the collected Q sets and the time needed (in milliseconds) by each combination of strategies. Although these tables give a general picture of the behaviour of the strategies, we consider that the results reflect what happens instance by instance. The detailed results of the experiments can be found in Appendix B.2.

In Table 3.1 it can be seen that the graph is contracted considerably by means of the shrinking, especially in large problems. The largest contractions are achieved with strategy S1S2. An interesting point of the results is that with the rules derived from Theorem 3.5 (C1,C2,C3) the support graph is simplified significantly, which encourages us to apply the shrinking preprocess for other valid inequalities, such as combs. Note that, rule C3 does not contract the graph more than what is already achieved by the combination of rules C2 and C3, see Section 3.6 for the discussion concerning this result.

Regarding the speedup up obtained by the shrinking strategies, the results are clear and show the importance of performing the shrinking preprocess before the separation algorithms. If we observe the column related to Algorithm EH in Table 3.1, the speedup obtained by each shrinking strategy is meaningful. In Medium instances, on average, the speedup is about 6 times for the least aggressive strategy (C1), and 17 times in Large instances. By means of the most aggressive strategy (S1S2) the speedup on average is 17 for Medium-sized instances and 53 in Large-sized instances.

With respect to the time needed, the separation algorithms, Algorithm DH and Al-

		Preprocess		Separation			
		Graph Size		Speedup			
Size	Shrinking	$\%\lvert V^*\rvert$	$\%\lvert E^*\rvert$	EH	DH	DHI	EPG
Medium	NO	100.00	100.00	1	9	9	9
	C1	42.55	50.61	6	29	23	19
	C1C2	39.73	46.40	7	32	27	20
	C1C2C3	39.73	46.40	7	33	25	20
	S1	22.88	26.43	16	57	51	28
	S1S2	21.26	24.53	17	60	53	27
Large	NO	100.00	100.00	1	15	15	16
	C1	30.45	37.88	17	107	74	139
	C1C2	27.95	34.10	20	122	86	151
	C1C2C3	27.95	34.10	20	121	80	150
	S1	16.15	19.91	44	221	203	215
	S1S2	14.34	17.43	53	252	227	225

Table 3.1: Average speedup of the proposed algorithms using the Algorithm EH with no shrinking preprocess as a baseline.

gorithm EPG, are both faster than the commonly used Algorithm EH, which shows the relevance of the detailed exact separation algorithms in Section 3.4. If we compare Algorithm DH and Algorithm EPG, without considering any shrinking strategy, the speedups on average are similar (9 and 9 times, respectively) and Algorithm EPG in larger instances (15 and 16 times, respectively). The table also suggests, based on the results of Algorithm DH and Algorithm DHI, that it is not convenient in the Dynamic Hong's separation algorithm to internally carry out extra shrinking procedures.

Speedup of SEC separation algorithms

Taking into account jointly the shrinking and separation strategies, the largest speedups are obtained when rules S1 and S2 are combined in the preprocess and, after that, alternatives to the standard Hong separation algorithms are used. In terms of running time, the Algorithm DH with the S1S2 shrinking preprocess obtains the best results in the experiments, with an average speedup of 60 in Medium-sized instances and 252 in Large-sized instances. The results obtained by Algorithm EPG with the S1S2 preprocess strategy are also outstanding, especially in large-sized instances with an average speedup of 225.

Apart from the running time, an aspect to consider when making a choice about the separation algorithm is the number of violated cuts found. As we have already mentioned, in the cycle problems, the number of collected violated SECs is closely related with the Q sets obtained by the separation algorithms. Therefore, we have measured the obtained amount of Q sets instead of the number of violated SECs. In Table 3.2, the average number of Q sets and time of each combination of strategies is shown.

The first aspect to note is that, by means of the shrinking preprocess, which is considerably faster than the exact separation procedure, we are able to find violated SECs in many instances (via Theorem 3.10 and Lemma 3.9). These violated SECs might be enough for the separation goal and, in practice, we could skip the exact separation algorithm if violated inequalities are found in the preprocess. In the separation process, in general, the largest amount of Q sets are obtained by Algorithm EPG, as was anticipated theoretically in Section 3.4. Note that, the quantity of obtained Q sets is sensitive to the randomness of the shrinking and separation strategies (it can be concluded because $\#Q$ is not always an integer).

In the view of these results, the S1S2 shrinking strategy is the best choice to use as the preprocess of SEC separation algorithms. Bearing in mind both the time and the obtained amount of Q sets, either Algorithm DH or Algorithm EPG might be a good choice as the separation algorithm. However, it is not clear from these results which of the two exact approaches should be used in practice. It probably depends on the nature and the size of the cycle problem under consideration.

		Preprocess		Separation							
		All		EH		DH		DHI		EPG	
Size	Shrinking	#Q	Time	#Q	Time	#Q	Time	#Q	Time	#Q	Time
Medium	NO	0.0	0.5	83.8	211.6	79.9	17.1	79.9	17.1	438.2	16.3
	C1	0.0	0.8	27.8	30.2	58.4	5.2	58.4	6.5	149.0	7.8
	C1C2	5.5	0.8	31.6	25.2	59.4	4.6	59.4	5.5	139.7	7.3
	C1C2C3	5.5	0.9	31.6	25.5	59.4	4.5	59.4	5.9	139.8	7.4
	S1	29.3	0.9	43.4	10.2	63.1	2.6	63.1	2.9	101.3	5.3
	S1S2	35.1	0.9	48.8	9.5	69.0	2.5	69.9	2.8	98.3	5.3
Large	NO	0.0	9.9	679.4	26578.2	372.6	2140.0	372.6	2140.0	3395.1	1828.8
	C1	0.0	22.5	154.2	1513.4	266.8	203.7	266.8	295.8	756.6	146.7
	C1C2	17.0	22.8	166.8	1320.0	271.7	179.3	271.7	257.2	717.9	135.2
	C1C2C3	16.8	23.2	166.6	1321.0	271.5	181.0	271.5	277.1	717.8	136.2
	S1	169.2	25.1	225.4	515.4	287.0	95.4	287.0	103.8	507.1	94.7
	S1S2	248.8	25.3	293.1	427.2	372.2	83.5	374.3	91.5	528.0	91.1

Table 3.2: On average, the number of Q sets found and the time needed by strategy and size.

3.6 Discussion

Finally, we would like to open a discussion about the following concerns as a consequence of the computational results. It might be helpful, to look at the detailed computational results in Appendix B.2 to understand the motivation behind the discussion below.

In Figure 3.9, an example of a vector $(y, x) \in P_A^G$ was shown where rule C3 can be applied but rules C1 nor C2 cannot. However, in the experiments, although rule C3 has been applied in some instances, we have not obtained any situation in which rule C3 was able to simplify the support graph more than with the rest of the rules. An open question is then to explain why rule C3 does not improve the results obtained by means of the rules C1 and C2. We believe that this is related with the planarity property of the support graphs, which is satisfied in the considered instances. Note that the graph in the example of Figure 3.9 is not planar because the complete graph of 5 vertices, K_5, is a subgraph of it.

Conjecture 3.21. *Given a graph G, let $(y, x) \in P_A^G$ be a vector. If the support graph G^* of (y, x) is planar, then the combination of the rules $C1$ and $C2$ dominate the rule $C3$.*

Note that the rules C1, C2, and C3 induce a contraction of an edge (a sequence of contractions for C3), which is a closed operation in planar graphs. Therefore, if G^* is planar then $G^*[S]$ is also planar for every subset S obtained from these rules. While working with the OP, we have empirically seen that in geometrical instances the support graph obtained within a B&C is planar most of the time.

Another interesting fact that can be extracted from the experiments is that the number of vertices and edges in the shrunk graph (the final result) is independent of the ordering of the considered rules and the shrinkable sets. This suggests the idea that the obtained shrunk graphs are isomorphic.

Conjecture 3.22. *Given a graph G, let $(y, x) \in P_A^G$ be a vector and SRK $\in$ {C1, C1C2, C1C2C3, S1, S1S2} be a fixed shrinking strategy, then the graphs obtained by applying SRK to (y, x) are isomorphic.*

If the conjecture is true, the complexity of the separation algorithm carried out in the shrunk graph does not depend on the different implementations of a shrinking strategy. As a consequence, in the future, we might focus on identifying the implementations of the shrinking strategies that might obtain the largest amount of Q sets, especially for the preprocess, e.g., by reordering the vertices in the heap.

3.7 Conclusions

In this chapter, for cycle problems, we have successfully generalized the global (C1, C2 and C3) and SEC specific (S1, S2 and S3) shrinking rules proposed in the literature of the TSP. The obtained computational results for the shrinking in the OP are remarkable and, hence, very promising for other cycle problems. The results clearly show that the shrinking technique considerably improves the running time of the separation algorithm for SECs. This opens the possibility to investigate in two directions in cycle problems: (1) studying the shrinking for other valid cycle inequalities of the OP (e.g., combs) and (2) evaluating for other cycle problems the shrinking technique in SEC separation problems.

Part of the chapter focuses on exact SEC separation algorithms for cycle problems. We have extended from the TSP two exact algorithms (Algorithm DH and Algorithm EPG). The proposed separation algorithms were shown to be more efficient in the OP than the exact algorithm used so far in the literature (the adaptation of the classical Hong's approach). The importance of the detailed extension of the Padberg-Grötschel approach, Algorithm EPG, lies in the fact that in cycle problems, in general, the global minimum cut of a support graph might not generate a violated SEC, while violated SECs in the same graph exist. An example is given where this claim is shown, which implies that the adaptions of the Padberg-Grötschel approach used so far in the literature of cycle problems should be viewed as heuristic separation algorithms. Therefore, this might be the first exact extension of the Padberg-Grötschel approach in the literature for cycle problems.

CHAPTER 4

RB&C: Revisited Branch-and-Cut Algorithm

Outline

In this chapter, we present an exact algorithm for the OP. These contributions deal with the separation algorithms of inequalities stemming from the cycle problem (SECs and comb inequalities), the design of the separation loop, the pricing of variables for the column generation and the calculation of the lower and upper bounds of the problem.

4.1 Introduction

The OP can be defined by a 5-tuple $\langle G, d, s, 1, d_0 \rangle$, where $G = K_n = (V, E)$ is a complete graph with vertex set V and edge set E; $d = (d_e)$ where d_e is the positive distance value (time or weight) associated to each $e \in E$; $s = (s_v)$, where s_v is a positive value that represents the score (profit) of vertex $v \in V$; $1 \in V$ is a vertex selected as the depot; and d_0 is a positive value that limits the cycle length.

The OP goal is to determine a simple cycle that maximizes the sum of the scores of the visited vertices, such that it contains the depot node $1 \in V$ and whose length is equal to or lower than the distance limitation, d_0. Then, the OP can be formulated as the following 0-1 Integer Linear model:

$$\max \quad \sum_{v \in V} s_v y_v \tag{4.1a}$$

$$\text{s.t.} \quad \sum_{e \in E} d_e x_e \leq d_0, \tag{4.1b}$$

$$x(\delta(v)) - 2y_v = 0, \qquad v \in V, \tag{4.1c}$$

$$x(\delta(H)) - 2y_l - 2y_r \geq -2, \qquad l \in H \subset V, r \in V - H, \quad 3 \leq |H| \leq |V| - 3\,, \tag{4.1d}$$

$$y_v - x_e \geq 0, \qquad v \in V, e \in \delta(v), \tag{4.1e}$$
$$0 \leq y_v \leq 1, \qquad v \in V, \tag{4.1f}$$
$$0 \leq x_e \leq 1, \qquad e \in E, \tag{4.1g}$$
$$y_1 = 1, \tag{4.1h}$$
$$x_e \in \mathbb{Z} \qquad e \in E \tag{4.1i}$$

where the objective function (4.1a) is to maximize the total collected profit. The constraint (4.1b) limits the total cycle length. The Subcycle Elimination Constraints (SEC) (4.1d) ensure that only one connected cycle exists. Throughout the chapter, we use the notation $\langle H, l, r\rangle$ for the SEC defined by the set $H \subset V$ and the vertices $l \in H$ and $r \notin H$. The constraints (4.1g) and (4.1i) impose that the edge variables are 0-1, consequently, considering these together with the Logical Constraints (4.1e) and the bounds (4.1g), the vertex variables are also 0-1. The constraint (4.1h) defines the depot condition.

As mentioned in the introduction, the OP can be seen as a combination of the TSP-decision and the KP problems. Particularly, the OP is a Cycle Problem (CP) where the solutions, which are cycles, need to satisfy a certain length constraint. This relation with the two classical optimization problems is useful when identifying the valid inequalities and their respective separation algorithms for OP. Let us show how the solution space of OP is related to those well-known problems. The OP Polytope (P_{OP}) of the complete graph K_n is defined by:

$$P_{OP} := conv\{(y,x) \in \mathbb{R}^{V\times E} : (y,x) \text{ satisfies } (4.1b), (4.1c), (4.1d), (4.1e), (4.1f), (4.1g), (4.1h), (4.1i)\} \tag{4.2}$$

The Knapsack Polytope (P_{KP}), see Balas [1975], is a well-studied polytope closely related to the P_{OP}:

$$P_{KP} := conv\{x \in \mathbb{R}^{E} : x \text{ satisfies } (4.1b), (4.1g), (4.1i)\} \tag{4.3}$$

Since the solutions of the OP are cycles, the Cycle Polytope (P_{CP}), presented in Chapter 3, plays a crucial role when solving the OP with B&C.

We have the following relationship:

$$P_{OP} \subset P_{CP} \cap (\mathbb{R}^V \times P_{KP}) \cap \{(y,x) \in \mathbb{R}^{V\times E} : y_1 = 1\} \tag{4.4}$$

Consequently, the potential valid inequalities for the OP are those which are valid for P_{CP} and the P_{KP}. However, the P_{OP} and the intersected polytopes in the relationship (4.4) are not equal and alternative valid inequalities are needed to deal with the OP. Figure 4.1 shows an example of a vector (y,x) in $P_{CP}\cap(\mathbb{R}^V \times P_{KP})\cap\{(y,x) : y_1 = 1\}$ but not in P_{OP}. Let G be the complete graph generated by the set $V = \{1,2,3,4,5\}$, and P_{OP} be the OP polytope of $\langle G, d, s, 1, d_0\rangle$ where d is the 2-dimensional euclidean distance determined by the numbers of the figure, s is any positive vector and the distance constraint

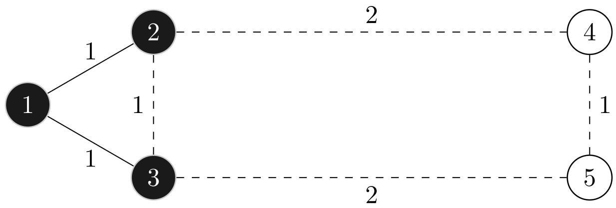


Figure 4.1: Example of a vector in $P_{CP} \cap (\mathbb{R}^V \times P_{KP}) \cap \{(y, x) : y_1 = 1\}$ but not in P_{OP}.

is set as $d_0 = 5$. The (y, x) vector is defined as follows: it is assumed that the degree equations are satisfied, and the dashed edges of figure have 0.5 value, 1 value the solid edges and 0 otherwise. On the one hand, the vector (y, x) belongs to P_{CP}. Consider the cycles $c_1 = ([1, 2], [2, 3], [1, 3])$ and $c_2 = ([1, 2], [2, 4], [4, 5], [3, 5], [1, 3])$ whose characteristic vectors, (y^{c_1}, x^{c_1}) and (y^{c_2}, x^{c_2}), belongs to P_{CP}. We have that (y, x) is a convex combination of the characteristic vectors of c_1 and c_2, i.e. $(y, x) = \frac{1}{2}(y^{c_1}, x^{c_1}) + \frac{1}{2}(y^{c_2}, x^{c_2})$. On the other hand, the vector x belongs to P_{KP}. Consider the sets $k_1 = \{[1, 2], [1, 3], [2, 3], [2, 4]\}$ and $k_2 = \{[1, 2], [1, 3], [3, 5], [4, 5]\}$ whose characteristic vectors, x^{k_1} and x^{k_2}, belong to P_{KP}. We have that x is a convex combination of the characteristic vectors of k_1 and k_2, i.e. $x = \frac{1}{2}x^{k_1} + \frac{1}{2}x^{k_2}$. However, (y, x) does not belong to P_{OP} since there is no cycle in P_{OP} containing node 4 (or node 5). It can be easily verified that a cycle containing 1 and 4 (or 5) and having at least three edges has a length strictly larger than 5.

4.2 Valid Inequalities

In this section, we present valid inequalities for the OP. The straightforward inequalities, as motivated in Section 4.1, are based on the P_{KP} (Edge Cover inequalities) and P_{CP} (Comb inequalities) relaxations of the P_{OP} and they were mainly proposed in Fischetti et al. [1998] and Gendreau et al. [1998b]. Additional valid inequalities to those based on P_{KP} and P_{CP} have also been proposed in the literature: the Connectivity Constraints in Leifer and Rosenwein [1994], the Vertex Cover inequalities in Gendreau et al. [1998b], and the Cycle Cover and the Path inequalities in Fischetti et al. [1998]. The novelty of this section is an alternative representation of comb inequalities, which is then used for the efficient pricing in Section 4.5.

4.2.1 Connectivity Constraints

The Connectivity Constraints (CC) are well-known inequalities for the OP, e.g. Gendreau et al. [1998b] and Leifer and Rosenwein [1994], and are a particular case of the conditional cuts proposed in Fischetti et al. [1998]. The CCs exploit the depot constraint (4.1h). Given a lower bound, LB, of the OP, let T be a subset of nodes such that $1 \in T$, $|T| \geq 2$

and $\sum_{v \in T} s_v \leq LB$. The inequality defined by T

$$x(\delta(T)) \geq 2 \tag{4.5}$$

is valid for the OP. Since $x(\delta(T)) = x(\delta(V - T))$, the inequality can also be defined for $T \subset V$ such that $1 \notin T$ and $\sum_{v \notin T} s_v \leq LB$. So, it is always possible to assume that $|T| \leq |V|/2$.

4.2.2 Comb Inequalities

The comb inequalities were generalized from the TSP to cycle problems in Bauer [1997]. A comb is a tuple $\langle H, \{T_1, \ldots, T_t\}, L, R\rangle$ of three vertex subsets and a family $\mathcal{T} = \{T_1, \ldots, T_t\}$ of vertex subsets such that satisfies the following properties:

i) $t \geq 3$ and an odd integer

ii) $T_i \cap T_j = \emptyset$ for $1 \leq i < j \leq t$

iii) $T_i \cap H \neq \emptyset$ and $T_i - H \neq \emptyset$ for $i = 1, \ldots, t$

iv) $L = \{l_i\}$ such that $l_i \in T_i \cap H$ for $i = 1, \ldots, t$

v) $R = \{r_i\}$ such that $r_i \in T_i - H$ for $i = 1, \ldots, t$

The set H is called the handle, the sets in $\mathcal{T}$ are called the teeth, the set R is called the *Root* set, and L is called the *Link* set. Then, the inequality

$$x(\delta(H)) + \sum_{j=1}^{t} x(\delta(T_j)) - 2y(R) - 2y(L) \geq 1 - t \tag{4.6}$$

is facet-defining for P_{CP}, as was shown in Bauer [1997], and therefore, a valid inequality for OP. When all the teeth consist of exactly two vertices, the comb inequalities are known as blossom inequalities.

4.2.3 Edge Cover Inequalities

The maximum length constraint (4.1b), which is a capacity constraint for the edge variables, defines a KP polytope, as explained in Section 4.1. For every feasible (y, x), the edge variable, x, belongs to P_{KP}. For the OP, the Edge Cover inequalities are the cover inequalities of the associated P_{KP} (Balas [1975]). These inequalities were first introduced for the OP in Leifer and Rosenwein [1994] and also used in Fischetti et al. [1998] and Gendreau et al. [1998b]. Let $F \subset E$ be a subset with $\sum_{e \in F} d_e > d_0$, then:

$$x(F) \leq |F| - 1 \tag{4.7}$$

defines an Edge Cover inequality for the OP. We assume that F is a minimal cover, i.e. for every $F_0 \subsetneq F$, we have $\sum_{e \in F_0} d_e \leq d_0$.

4.2.4 Cycle Cover Inequalities

Every feasible cycle $F \subset E$ satisfies the equation $x(F) = y(V(F))$. Let $F \subset E$ be a subset that defines a cycle with $\sum_{e\in F} d_e > d_0$, then the inequality

$$x(F) \leq y(V(F)) - 1 \tag{4.8}$$

is valid for the OP. These cuts were used in Fischetti et al. [1998] and Gendreau et al. [1998b].

4.2.5 Vertex Cover Inequalities

Let UB be an upper bound of the OP and $Q \subset V$ be a subset with $\sum_{v\in Q} s_v > UB$, then:

$$y(Q) \leq |Q| - 1 \tag{4.9}$$

defines a Vertex Cover inequality for the OP. We assume that S is a minimal cover. These inequalities were first used for the OP in Gendreau et al. [1998b].

4.2.6 Path Inequalities

The goal of these cuts is to exclude the paths that due to the length constraint (4.1b) cannot be part of a feasible solution. Let $P = \{[i_1, i_2], [i_2, i_3], \dots, [i_{k-1}, i_k]\}$ be any simple path through $V(P) = \{i_1, \dots, i_k\} \subset V - \{1\}$, and define the vertex set:

$$W(P) := \{v \in V - V(P) : d_{1,i_1} + \sum_{e\in P} d_e + d_{i_k,v} + d_{v,1} \leq d_0\} \tag{4.10}$$

Then the following Path inequality

$$x(P) - y(V(P)) + y_1 + y_k - \sum_{v\in W(P)} x_{i_k,v} \leq 0 \tag{4.11}$$

is valid for the OP, see Fischetti et al. [1998].

In Figure 4.2 a flowchart representing a simplified B&C algorithm can be consulted.

4.3 Initialization

First of all, we obtain an initial heuristic solution. To that aim, we make use of the EA4OP metaheuristic in Kobeaga et al. [2018] considering a small size population.

Next, we build the initial subproblem, LP_0. Given the computational requirements of considering all the variables and constraints that define the OP, an initial subproblem LP_0 is built. The LP_0 is initialized considering the following subset of constraints and variables:

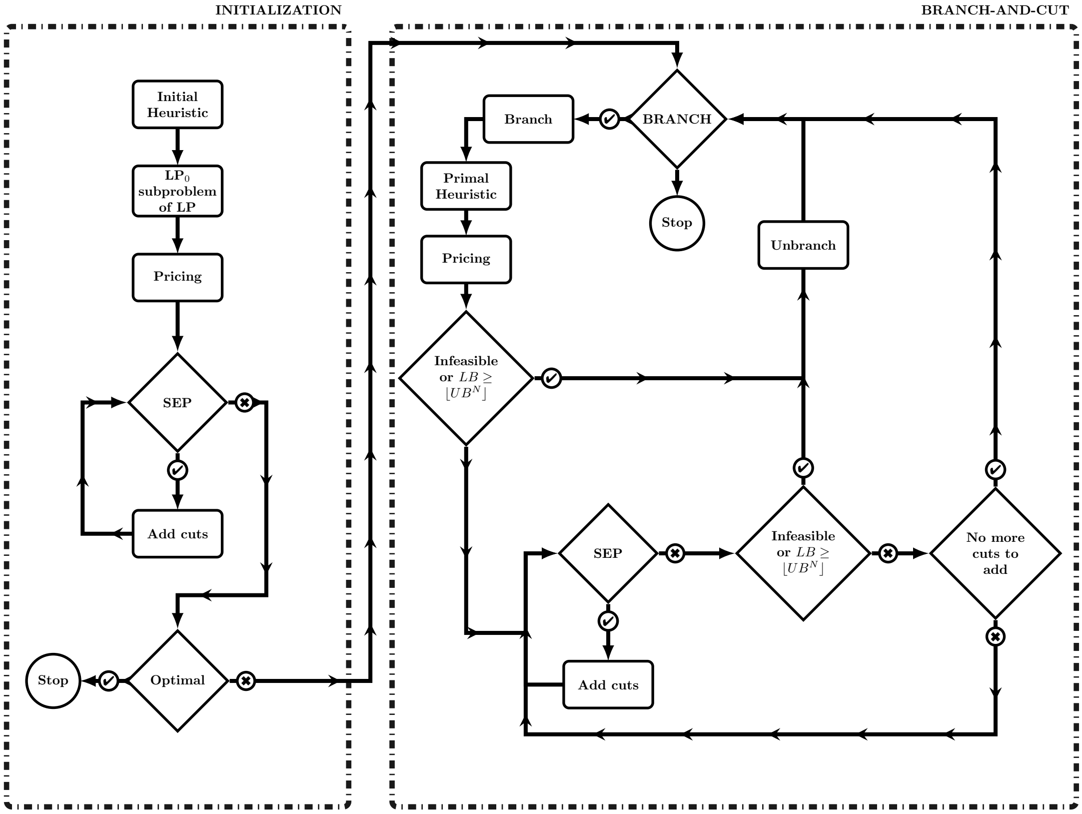


Figure 4.2: Flowchart of the Branch-and-Cut algorithm considered in this work. BRANCH is an oracle which returns an unevaluated node in the branching tree. SEP refers to the separation algorithms. At each action box of the flowchart the subproblem LP_0 is updated and solved.

i) All the vertex variables.

ii) Edges in the 10 nearest neighborhood graph.

iii) Maximum length constraint (4.1b), degree constraints (4.1c), and depot constraint (4.1h).

iv) Variable bounds, (4.1f) and (4.1g).

Immediately after the initialization, the edge variables are priced, see Section 4.5. In the rest of the chapter, we use the LP_0 symbol to refer to any subproblem of the OP, regardless of whether it is the initial one or not.

4.4 Separation algorithms

In this section, we present the heuristic and exact separation algorithms used to find the violated inequalities. Our contributions are concentrated in the separation algorithms for SECs, CCs and blossom inequalities. Hence, we only give details of these separation algorithms in the section. The details of separation algorithms for the rest of the inequalities (Logical Constraints, Edge Cover, Vertex Cover, Cycle Cover, and Path inequalities) can be found in Fischetti et al. [1998].

In this section, we present the heuristic and exact separation algorithms used to find the violated inequalities. Our contributions are concentrated in the separation algorithms for SECs, CCs and blossom inequalities. Hence, we only give details of these separation algorithms in the section. The details of separation algorithms for the rest of the inequalities (Logical Constraints, Edge Cover, Vertex Cover, Cycle Cover, and Path inequalities) can be found in Fischetti et al. [1998].

Let (y^*, x^*) be a solution of a particular LP_0 problem and define $V^* = \{v \in V : y_v^* > 0\}$ and $E^* = \{e \in E : x_e^* > 0\}$. Then, $G^* = (V^*, E^*)$ is called the support graph associated with the solution (y^*, x^*).

4.4.1 SECs and CCs

Violated SECs (4.1d) and CCs (4.5) are found using a common separation algorithm. This is natural since, in both constraint families, the incidence vector of the arcs, x in the inequality can be written as the star-set value, $x(\delta(Q))$ of a subset Q of vertices. Since $\delta(Q)$ is the cut associated with Q, the separations of both inequalities are closely related to the minimum cut problem. In Kobeaga et al. [2020a] it was shown that the shrinking techniques substantially speed up the SEC separation algorithms. However, as explained below, the shrinking might also have a negative impact on the finding of violated CCs. In this section, we study how to efficiently use the shrinking to speed up the joint separation algorithm by reducing the adverse effects for CCs.

Given a solution (y^*, x^*) and a subset Q, the subset Q could generate at the same time a violated SEC and a violated CC for (y^*, x^*). Since the CCs do not depend on the value of the vertices, while the SECs do, the CCs tend to be more violated and more stable, i.e., remain active in subsequent updates of the LP_0, than the SECs. Therefore, we treat the CCs with a higher priority.

Although SECs are part of the OP model, in order to control the size of the working LP_0, they are included only when required. This strategy is reasonable since there exist polynomial exact separation algorithms for SECs. In contrast, the separation problem for CCs is not known to be polynomial, and it can be modeled as follows:

$$\min \quad 2\sum_{v\in V^*} y_v^* z_v - 2\sum_{v\in V^*} x_{(v,u)}^* z_v z_u \tag{4.12a}$$

$$s.t: \quad \sum_{v\in S} s_v z_v \le LB \tag{4.12b}$$

$$z_1 = 1 \tag{4.12c}$$

$$z_v \in \{0,1\} \qquad \forall v \in V \tag{4.12d}$$

where $z = (z_v)$ are binary variables whose values are $z_v = 1$ if the node v is selected and 0 otherwise. The problem (4.12) is a Quadratic Knapsack Problem (QKP) with a fixed variable. Consequently, there exists a violated CC for (y^*, x^*) if and only if the optimal solution of Problem (4.12) has a value lower than 2. Taking into consideration that repeatedly solving QKPs during the B&C is not viable, the CCs are not separated exactly, but in a heuristic manner take advantage of the SEC separation algorithm. The well-known approaches for the separation of SECs in the TSP, the connected component heuristic and Hong's approach can be extended to jointly separate the SECs and CCs:

Connected components heuristic. The straightforward heuristic to find violated SECs and CCs is to search for the connected components of G^* using the depth-first-search algorithm. When a connected component contains the depot vertex 1 and the sum of the vertices scores in the component is lower than LB, we record the associated CC of the component, otherwise, we record the associated SECs.

Extended Hong's approach. There are two main strategies to exactly separate SEC inequalities in cycle problems, which are extensions of Hong's approach and the Padberg-Grötschel approach (also known as the Gomory-Hu tree-based approach) for the TSP, see Kobeaga et al. [2020a]. In both approaches, the separation is carried out by solving a sequence of $|V^*| - 1$ (s,t)-minimum cut problems. On the one hand, in the extended Hong's approach, the vertex with a higher y^* value (the depot vertex 1) is fixed to be the source, s, and the sink vertices, t, are chosen from the set $V^* - \{1\}$. On the other hand, the extended Padberg-Grötschel approach is based on the so-called Gomory-Hu tree (directed and rooted in 1), which is constructed by solving $|V^*| - 1$ (s,t)-minimum cut problems.

As mentioned above, and as already proposed in the literature, the SEC separation strategies are leveraged to find violated CCs as well. Although the extended Padberg-Grötschel approach obtains a larger number of violated SECs, it is not appropriate to find violated CCs, since the obtained sets do not contain the depot vertex 1. Contrarily, the extended Hong's approach for SECs can be easily adapted to additionally find violated CCs. It can be achieved, by solving at each step of the separation algorithm the $(1,v)$-minimum cut (useful to find violated SECs) and $(v,1)$-minimum cut (useful to find violated CCs) problems. For these reasons, we use the extended Hong's approach as the base strategy for the joint separation algorithm.

The running time of these SEC separation algorithms can be improved using the shrinking techniques for cycle problems, as was seen in Kobeaga et al. [2020a]. In this publication, three general shrinking rules (C1, C2, and C3) and three SEC specific shrinking rules (S1, S2, and S3) for cycle problems were presented. However, although the shrinking is a key strategy for efficiently separating the SECs, it might be unfavorable for the separation of CCs. The point is that when the vertices are contracted and grouped, the chance to obtain the subset of vertices with a score sum lower than LB decreases, consequently, some violated CCs might vanish. Note that, the mentioned shrinking techniques are safe for valid inequalities of the cycle polytope and CCs are not. Therefore, since CCs are important cuts for OP, shrinking might have a negative impact on the performance of the overall B&C algorithm for the OP. One contribution in this chapter is to propose strategies to minimize the possible disadvantages of the shrinking (which is important to speed up the separation) in the joint separation algorithm for SECs and CCs.

Following this, not all the shrinking strategies for cycle problems described in Kobeaga et al. [2020a] are adequate for the OP problem. Particularly, we exclude the S2 shrinking rule (which leads to excessively aggressive shrinking strategies and hence to vanish violated CCs in some cases) and only consider the shrinking strategies C1C2 and S1 in the preprocess of the joint separation algorithm. Once entered in the separation algorithm, the shrinking rule S3, which contracts the sink and target of the solved minimum cut, contributes positively to separating both families of constraints since it enables a wider family of subset candidates to be obtained. Hence, the S3 rule is used in combination with the C1C2 and S1 shrinking strategies in the separation algorithm. After the S3 rule is applied, we search for new shrinkable sets using the selected shrinking strategy.

Classically, the candidate subsets for SECs and CCs are obtained by the minimum cut algorithm. However, considering the importance of CCs, we intensify the search for extra candidate subsets for CCs, which is made more efficient by taking advantage of the vertex clustering obtained by the shrinking. We propose new strategies based on the following lemma:

Lemma 4.1. *Let (y, x) be a vector that satisfies the degree constraints. If U and W are subsets of V such that $W \subset U$, the following inequality is satisfied:*

$$x(\delta(U - W)) \leq x(\delta(U)) + x(\delta(W)) \tag{4.13}$$

Proof. When (y, x) satisfies the degree constraints, the identity $x(\delta(T)) = 2y(T) - 2x(E(T))$ is valid for every $T \subset V$. Replacing the respective expressions in the inequality (4.13) we obtain:

$$2y(U - W) - 2x(E(U - W)) \leq 2y(U) - 2x(E(U)) + 2y(W) - 2x(E(W))$$

Considering the hypothesis $W \subset U$, we have $y(U - W) = y(U) - y(W)$.

$$x(E(U)) - x(E(U - W)) \leq 2y(W) - x(E(W)) \tag{4.14a}$$

Also, if $W \subset U$, the equality $E(U - W) = E(U) - E(W) - \delta(W) \cap E(U)$ holds.

$$\begin{aligned} x(E(U)) - x(E(U)) + x(E(W)) + x(\delta(W) \cap E(U)) &\leq 2y(W) - x(E(W)) \\ x(\delta(W) \cap E(U)) &\leq 2y(W) - 2x(E(W)) \\ x(\delta(W) \cap E(U)) &\leq x(\delta(W)) \end{aligned}$$

This last inequality is satisfied due to $\delta(W) \cap E(S) \subset \delta(W)$, which proves the lemma. □

Let $G[S] = (V[S], E[S])$ be the graph and $(x[S], y[S])$ the vector obtained by applying a shrinking strategy to G^* and (y^*, x^*), respectively, and $\pi : \mathcal{P}(V[S]) \rightarrow \mathcal{P}(V)$ the unshrinking function. Let $\bar{Q}$ be the subset obtained by the $(\bar{v}, \bar{1})$-minimum cut (where $\bar{1}$ is the contracted vertex such that contains the depot vertex 1, i.e. $1 \in \pi(\bar{1})$), so $1 \in \pi(\bar{Q})$, and suppose that $x(\delta(\bar{Q})) < 2$. Note that, $x(\delta(Q)) = x[S](\delta(\bar{Q}))$, where $Q = \pi(\bar{Q})$. If $\sum_{v \in Q} us_v \leq LB$, the subset Q defines a violated CC. Otherwise, after each $(\bar{v}, \bar{1})$-minimum cut problem is solved, and in the case that $x(\delta(\bar{Q})) < 2$, we test the following strategies to find candidate subsets for CCs:

i) First, when $|\pi(\bar{1})| > 2$, we check if $y[S](\bar{1}) < 1$ and $\sum_{v \in \pi(\bar{1})} s_v \leq LB$. If this is the case, the subset $Q = \pi(\bar{1})$ defines a violated CC.

ii) Then, we check if there exists $\bar{v} \in \bar{V} - \bar{1}$, such that $x[S](\delta(\bar{Q})) + 2y[S](\bar{v}) < 2$ and $\sum_{v \in \pi(\bar{Q} - \bar{v})} s_v \leq LB$. If both inequalities are satisfied for $\bar{v}$, the subset $\pi(\bar{Q} - \bar{v})$ defines a violated CC.

iii) Finally, we sort the vertices in $\bar{Q} - \bar{1}$ in non-decreasing order of $\bar{y}$, and check greedily for the greatest subset $Q^{'} = \{\bar{v_1}, \ldots, \bar{v_k}\}$ of $\bar{Q}$ such that $x[S](\delta(\bar{Q})) + 2\sum_{v \in Q'} y[S](\bar{v}) < 2$. If $\sum_{\bar{v} \in \pi(\bar{Q} - Q')} s_v \leq LB$, the subset $\pi(\bar{Q} - Q^{'})$ defines a violated CC.

4.4.2 Comb Inequalities (blossoms)

For the B&C presented in this work, we only use the blossom subfamily of comb inequalities. In this section, we present two heuristics to search for violated blossom inequalities in cycle problems, and in particular, for the OP. The heuristics are extensions of the Padberg and Hong [1980] and Grötschel and Holland [1991] separation algorithms, developed in the context of the TSP.

The key point of the heuristics for blossom inequalities is to identify a subset of candidate handles to restrict the search of violated blossoms. In the literature of OP, a heuristic to find handle candidates is detailed in Fischetti et al. [1998]. In this heuristic, the search is guided by the greedy algorithm of Kruskal for the Minimum Spanning Tree. At each iteration of the Kruskal algorithm, a new edge is inserted into the tree, and the connected component containing the edge is chosen as a candidate handle. In this work, we consider two alternative approaches to finding candidate handles: the Extended Padberg-Hong heuristic and the Extended Grötschel-Holland heuristic.

Extended Padberg-Hong heuristic (EPH). Padberg and Hong [1980] proposed a blossom separation heuristic for the TSP, which is known as the odd-component heuristic. In this heuristic for the TSP, the violated blossoms are found by restricting the set of candidate handles to the connected components of the fractional graph $G_1^* = (V_1^*, E_1^*)$, where $E_1^* = \{e \in E^* : 0 < x_e^* < 1\}$ and $V_1^* = V(E_1^*)$.

Extending blossom heuristic for cycle problems

We generalize the blossom heuristic for the general cycle problems by applying the algorithm by levels. A level, λ, is defined by each different value of the set $\{y_v^*\}_v$. We call L the set of distinct levels. Note that, the number of levels, $|L|$, is bounded by $|V|$. Associated with a level we have the level graph $G_\lambda^* = (V_\lambda^*, E_\lambda^*)$, where $E_\lambda^* = \{e \in E^* : 0 < x_e^* < \lambda\}$ and $V_\lambda^* = V(E_\lambda^*)$.

A faster heuristic to find the handle candidates can be designed by omitting some connected components of G_λ^*. At every level, λ, we discard the connected components, C_i^λ, such that $y_v \neq \lambda$ for all $v \in C_i^\lambda$. Now, we identify the connected component of vertices with $y_v = \lambda$. So, in total, we search for $|V^*|$ different connected components of, in the worst case, G_1^*.

Once we have identified an initial list of candidate handles, the next step is to find the associated teeth for these handles. Let H be a candidate handle, and define the set of teeth as $\mathcal{T}_H = \{e \in \delta(H) : x_e^* \geq \lambda\}$. Recall that the teeth of blossoms are edges. Not all the teeth families obtained using this strategy satisfy the comb (blossom) definition. If two teeth overlap in $v \notin H$, then these two teeth are removed from the family of teeth $\mathcal{T}_H$ and the handle is updated as $H = H \cup \{v\}$. If, eventually, the list of teeth $\mathcal{T}_H$ consists of an odd number of at least three disjoint teeth, $\langle H, \mathcal{T}_H, L, R\rangle$ forms a blossom inequality where $L_i = T_i^j \cap H$ and $R_i = T_i^j - H$. If there is just one tooth i.e., $\mathcal{T}_H = \{T\}$, we test if H defines a violated CC. In the case that it does not, then H alone defines a violated SEC.

Extended Grötschel-Holland heuristic (EGH). Another fast heuristic for the TSP was proposed in Grötschel and Holland [1991] whose aim was to minimize the influence of small perturbations of x^* in the separation algorithm. We have adapted this heuristic for the OP using the strategy of levels mentioned above. In this approach, the handles are considered as the vertex sets of the connected components of the graph $G_{\lambda,\epsilon}^* = (V^*, E_{\lambda,\epsilon}^*)$ where

$$E_{\lambda,\epsilon}^* = \{e \in E_\lambda^* : \epsilon \leq x_e^* \leq (1-\epsilon)\lambda\}$$

for a small ϵ, $0 < \epsilon < 1$. Let H denote the vertex set of such a component, a candidate handle, and let $e_1, \ldots, e_t$ be the edges in the set

$$\mathcal{T}_H = \{e \in \delta(H) \subset E^* : x_e^* > (1-\epsilon)\lambda\}$$

in the non-increasing order of x_e^*. If t is even, then append to $\mathcal{T}_H$ the edge with the highest x_e^* in

$$\{e \in \delta(H) \subset E^* : x_e^* < \epsilon\}$$

If the edges intersect, the strategy outlined above is followed to obtain a handle H and a teeth family $\mathcal{T}_H$ that satisfies the blossom definition.

In Figure 4.3 we illustrate the EPH blossom heuristic for cycle problems. In Figure 4.3.a) the given support graph is presented, where there are three distinct levels, $L =$

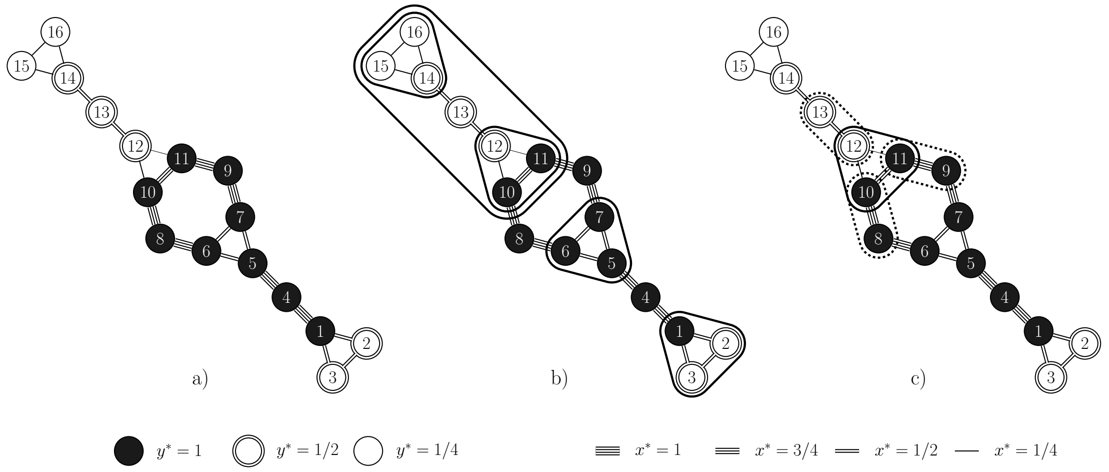


Figure 4.3: Illustration for the Extended Padberg-Hong blossom heuristic. Figure a) represents the support graph, with the vertex and edge values detailed in the bottom legend. Figure b) shows all the handle candidates obtained by the heuristic. Figure c) a violated blossom found by the heuristic involving vertices with different y values.

$\{1, 1/2, 1/4\}$. In Figure 4.3.b) the candidate handles are presented. Three candidate handles are obtained in level 1: $\{1, 2, 3\}$, $\{5, 6, 7\}$ and $\{10, 11, 12, 13, 14, 15, 16\}$. Two candidate handles are obtained in level 1/2: $\{10, 11, 12\}$ and $\{14, 15, 16\}$. There are no candidate handles obtained in level 1/4. Next, we check for violated cuts. The star-set of $\{10, 11, 12, 13, 14, 15, 16\}$ is formed by two non-overlapping edges, so it is excluded. The candidates $\{5, 6, 7\}$ and $\{10, 11, 12\}$ define violated blossoms, e.g., $\langle\{10, 11, 12\}, \{\{8, 10\}, \{9, 11\}, \{12, 13\}\}, L, R\rangle$ where $L = \{10, 11, 12\}$ and $R = \{8, 9, 13\}$ shown in Figure 4.3.c). The candidates $\{1, 2, 3\}$ and $\{14, 15, 16\}$ define violated SECs, e.g. $\langle\{1, 2, 3\}, 1, 4\rangle$ and $\langle\{14, 15, 16\}, 14, 1\rangle$, but first for $\{1, 2, 3\}$ it should be checked whether it defines a violated CC.

4.5 Column Generation

During the B&C algorithm, only a subset of edges is included in the working LP_0. At certain points of the algorithm, we need to price the excluded edge variables, and add to the LP_0: 1) to guarantee that the working relaxation is an upper bound of the problem or branched subproblem and 2) to recover, whenever it is possible, a feasible LP_0 after feasibility breaking cuts have been added to the LP_0. Taking into account that usually only a small subset of variables is included in the LP_0, and that the excluded variables could participate in multiple cuts of the LP_0, the pricing phase could constitute a bottleneck of the B&C algorithm. In this section, we develop a technique, inspired by that used in Applegate et al. [2007], which enables us to avoid repetitive calculations and to skip the exact calculation of the reduced cost of some variables.

Let us call $\mathcal{L}^V$ the family of SECs (4.1d), CC (4.5), and comb (4.6) cuts. In these cuts, the edge variables with non-negative coefficients can be represented as the sum star-set of subsets of vertices. Complementarily, let us call $\mathcal{L}^E$ the family of Logical (4.1e), Edge Cover (4.7), Cycle Cover (4.8) and Path (4.11) cuts. Note that the Vertex Cover (4.9) inequalities do not contribute to the reduced cost of the edge variables. So, in the OP, the reduced cost of an edge variable, $e = [v, w]$, can be calculated by:

$$rc_e = -d_e\pi_{d_0} - \pi_v - \pi_w + rc_e^V + rc_e^E \tag{4.16}$$

where π_{d_0} is the dual variable of the maximum length constraint (4.1b), π_v and π_w are the dual variables of the degree constraints (4.1c) of v and w respectively, and rc_e^V and rc_e^E are the contributions made by the cuts in $\mathcal{L}^V$ and $\mathcal{L}^E$, respectively. We will see that the rc_e^E values can be obtained in linear time in terms of $|V|$ and $|\mathcal{L}^E|$, and we will reproduce the pricing strategy used in Applegate et al. [2007] to calculate the rc_e^V values.

It can be seen that the cost of the calculation of all the rc_e^E is $O(|\mathcal{L}^E||V|)$. To that aim, it is sufficient to check that the number of edges with a non-negative coefficient in each cut of $\mathcal{L}^E$ is bounded by $|V|$. In the case of Logical, Cycle Cover, and Path inequalities, it is derived from the definition of the valid inequality. For Edge Cover inequalities, this bound is obtained in Lemma 4.2.

Lemma 4.2. *Let $T \subset E$ denote a subset defining a violated cover inequality. If the degree equations* (4.1c) *are satisfied by $(y, x) \in \mathbb{R}^{V \times E}$ then $|T| \leq |V|$.*

Proof. When the degree constraints are satisfied by (y, x), as a consequence of the well-known equality $x(\delta(S)) = 2y(S) - 2x(E(S))$, the inequality $x(E(V(T))) \leq y(V(T))$ is always satisfied. Suppose that T violates the cover inequality (4.7) then

$$|T| - 1 < x(T) \leq x(E(V(T))) \leq y(V(T)) \leq |V| \tag{4.17}$$

□

Calculating all the rc_e^V values has a $O(|\mathcal{L}^V||V|^2)$ complexity when the cuts are stored externally as edge variable coefficient arrays. The strategy used in Applegate et al. [2007] speeds up the pricing by obtaining a fast lower bound of the reduced cost rc_e^V (TSP is a minimization problem) and excluding for exact pricing the edges that have a negative lower bound. In order to use this strategy for the OP, first, the edge variables of the cuts in $\mathcal{L}^V$ must be represented and stored as a family of subsets of vertices, as we have done in Section 4.2. Let $\mathcal{S} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_r$ be the family of all the subsets involved in the cuts of $\mathcal{L}^V$ where $\mathcal{F}_i = \{H_i\} \cup \mathcal{T}_i$. For combs, H_i and $\mathcal{T}_i$ represent the handle and teeth set, respectively. For SECs and CCs we can assume that $\mathcal{T}_i = \emptyset$ and $H_i = \emptyset$, respectively.

Based on the representation of the cuts in $\mathcal{L}^V$ by means of subsets of vertices, the cuts are stored in an efficient data structure by pointing to the subsets involved in the cut.

This way each subset is saved once at most for all the cuts. Moreover, it allows us to speed up the evaluation of rc_e^V values as explained below.

Since the OP is a maximization problem, during the pricing, we need to identify the edge with positive reduced cost. We aim to define upper bounds, $\hat{rc}_e$, of the reduced costs rc_e, to exclude for exactly pricing the edges that have a non-positive upper bound $\hat{rc}_e^V$.

For each subset, $S \in \mathcal{S}$, let us call π_S the dual of the subset S defined as:

$$\pi_S = \sum_{j=1}^{r} \chi_j(S)\pi_j \tag{4.18}$$

where $\chi_j(S) = 1$ if $S \in \mathcal{F}_j$ and 0 otherwise, and π_j is the dual variable associated with the cut j. Then, the contribution of the cuts in $\mathcal{L}^V$ in the reduced cost of an edge e can be written as:

$$rc_e^V = \sum_{\substack{S \in \mathcal{F} \\ V(e) \cap S \neq \emptyset \\ V(e) - S \neq \emptyset}} \pi_S \tag{4.19}$$

where π_S is the dual of a subset S. Since, for the edge $e = [v, w]$, each S must contain either v or w, an upper bound, $\hat{rc}_e^V$, of rc_e^V can be obtained by:

$$\hat{rc}_e^V = \sum_{\substack{S \in \mathcal{F} \\ v \in S}} \pi_S + \sum_{\substack{S \in \mathcal{F} \\ w \in S}} \pi_S$$

which satisfies $rc_e^V \leq \hat{rc}_e^V$. Therefore, we have the desired upper bound:

$$\hat{rc}_e = -d_e\pi_{n+1} - \pi_v - \pi_w + rc_e^E + \hat{rc}_e^V \tag{4.20}$$

Pricing: cost and strategy

Note that, each edge appears at most twice in a comb inequality, so the calculation of all the $\hat{rc}_e^V$ has a $O(M|\mathcal{L}^V||V|)$ time complexity where M is the maximum number of subsets involved in a cut. Therefore, the calculation of all the $\hat{rc}_e$ has a $O(M|\mathcal{L}^V||V|)$ time complexity. In our B&C, the value of M is related to the number of teeth in the combs. To ensure a true linear time complexity procedure, one could limit the number of teeth in the combs. However, in practice, the number of teeth tends to be small and it can be assumed that $M << |V|$. The edges that $\hat{rc}_e \leq 0$ can be excluded for exactly pricing .

For those edges that $\hat{rc}_e > 0$, the exact reduced cost, rc_e, can be calculated by using the upper bound value:

$$rc_e = \hat{rc}_e - 2 \sum_{\substack{S \in \mathcal{F} \\ V(e) \subset S}} \pi_S \tag{4.21}$$

The pricing loop is done in batches. In the first step, a fixed number of $\hat{rc}_e$ are calculated, the first batch of variables and those with positive values are preselected. In the next step, for those preselected variables, we calculate the exact reduced cost, rc_e, and add to the LP_0 the edges whose value is positive. Then, the LP_0 is updated. Next, we select the second batch of variables and we repeat the procedure. When the pricing aims to obtain the upper bound of the branched subproblem, we exit the pricing loop when a whole round of evaluation is performed without introducing a variable to the LP_0. When the pricing aims to recover a feasible LP_0, we exit the pricing loop once a feasible LP_0 is obtained without the need to price all the excluded variables.

4.6 Separation Loop

The separation loop to find the violated cuts is accomplished in three subloops. In the inner loop, we consider the separation of logical constraints (4.1e) and the connected components heuristic for SECs and CCs. In the middle loop, we consider the separations of cuts which are related to the cycle essence of the OP, i.e., SECs, CCs, blossoms, and Cycle Cover cuts. In the outer loop, we consider the rest of the cuts, i.e., the Edge Cover, Vertex Cover, and the Path inequalities. The separation loop is illustrated in Figure 4.4.

At each subloop, the separation of the considered cuts is performed sequentially, instead of restarting from the beginning of the list. This is, we always carry out the next separation in the subloop list, regardless of whether or not we are coming from an interior subloop. This way, we give the same chance to all separations in a subloop and decrease the probability of bounding in the same separation algorithm in consecutive iterations of the subloop.

The separation algorithms of the inner loop are fast since both have a $O(|E^*|)$ time complexity. First, we carry out the connected components heuristic and then the separation of logical constraints. In the inner loop, intending to keep it as a fast loop, we price the edge variables only when the floor part of the objective value is equal to the lower bound of the OP, i.e., if $\lfloor s \cdot y^* \rfloor = LB$. When both separations fail and no new edges have been added, we find a feasible solution using the PB primal heuristic (see Section 4.7) and update the LB if needed. We add the associated CC of the heuristic solution if it is violated and then we price the variables. When a new CC cut or a priced edge has been added to the LP_0, the inner loop is repeated. Otherwise, we return to the middle loop.

The middle and outer loops only differ in the considered constraint families. In the middle loop, we consider the separation algorithms in the following order: the extended Padberg-Hong algorithm for blossom, the extended Grötschel-Holland algorithm for blossom, the joint SEC/CC separation algorithm, and Cycle Cover separation algorithm. In the outer loop, we consider the Edge Cover algorithm, the Vertex Cover algorithm, the Path algorithm.

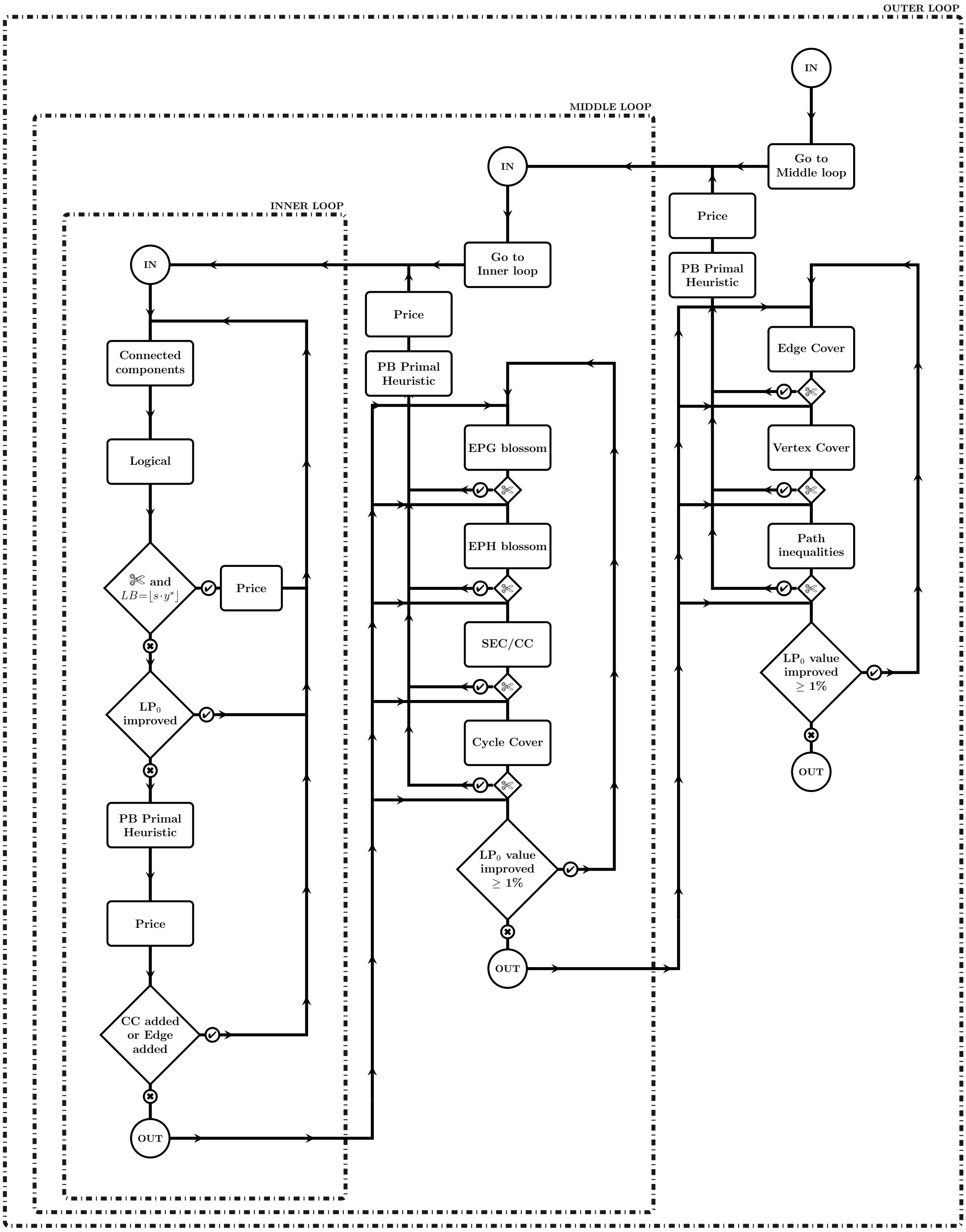


Figure 4.4: Illustration of the separation loop. The symbol ✄ represents that some cuts have been added to the LP_0.

Separation loop strategy

When we enter in any of the separation loops, the first step is to execute the lower level subloops. Then, we start with the first algorithm on the list. If no violated cuts are found we move on to the next algorithm. If violated cuts are found, we first add the cuts and optimize the LP_0. Then, we search for a feasible solution using a primal heuristic and update the LB if needed. We add the associated CC of the heuristic solution in case it is violated and then we price the variables. At this point, we move to the lower level loop and continue with the next separation in the list.

In the separation loop, after adding the violated cuts found in a separation algorithm, we check if any edge variable or constraint can be removed from the LP_0. We remove an edge variable from the LP_0 if, during a number of consecutive evaluations, its associated value, x_e^*, has been zero. We remove a constraint from the LP_0 if during a number of consecutive evaluations its slack has been higher than zero.

4.7 Primal Heuristics and Lower Bounds

We use two primal heuristics to obtain feasible solutions from a fractional solution (y^*, x^*). In the first heuristic, we obtain a single solution, by using the x^* values related to edges, inspired by the heuristic proposed in Fischetti et al. [1998]. In the second heuristic, first, we build a population of cycles and then evolve it using the EA4OP metaheuristic, see Kobeaga et al. [2018]. The cycles in the population are constructed by selecting first the subset of vertices in each cycle using the y^* values.

Path Building primal heuristic (PB). The PB heuristic was presented in Fischetti et al. [1998]. First, the edges $e \in E^*$ are sorted in decreasing order of x_e^*, and the ties are randomly broken. The procedure starts with an empty path $T = \emptyset$. At each step we select an edge $e \in E^*$ whose x_e^* has the largest value from the set of edges which have not been considered yet. If the inclusion of e in T does not lead to a vertex with a degree larger than 2, then $T = T \cup \{e\}$ otherwise we exclude e and repeat the process. The path building heuristic finishes when the inclusion of e leads to T being a cycle or when there are no edges left to check. If the depot vertex is not in one of the paths in T, it is included as a single point path. If T consists of multiple paths, we extend it to a cycle by randomly connecting the extreme vertices (in the original paper the paths were joined using the nearest neighbor heuristic). Since this primal heuristic is fast, it is used in the separation loop.

Vertex Picking primal heuristic (VP) with the EA4OP metaheuristic. In the VP heuristic, we first select a collection of vertices in V^* and then build a random cycle through the selected vertices. Each vertex v is selected according to a Bernouilli distribution with parameter y_v^*. By applying multiple times the VP strategy to obtain feasible solutions from (y^*, x^*), we build a small population. Then, as explained below, we ensure that the solutions in the population are feasible and improve when it is possible. Once we have a population with feasible solutions, it is evolved using the EA4OP metaheuristic proposed in Kobeaga et al. [2018]. The EA4OP with VP heuristic is used to find feasible solutions after an edge is branched, as shown in Figure 4.2.

For solutions obtained by PB and VP heuristics, we improve the route lengths using the Lin-Kernighan heuristic for the TSP, and then first check if it satisfies the constraint (4.1b). If it does not, we apply the drop operator which consists in deleting vertices from the solution until the cycle satisfies the length constraint. Then we try to improve the solution by the k-d tree based vertex inclusion procedure as explained in Kobeaga et al. [2018].

4.8 Branching and Upper Bounds

The branching is carried out in a classical way following a depth-first-search, where the edges are branched first to 1 and then to 0. In order to select the edge variable to branch, we use the classical branching strategy: the edge e, with the fractional value closest to 0.5 is selected, i.e., the edge that minimizes $|x_e^* - 0.5|$.

Global and branch node upper bounds

The global upper bound and branch node upper bound are calculated just before pruning a branch. The branch node upper bound, UB^N, is used to verify the pruning, i.e, that $LB \geq \lfloor UB^N \rfloor$. The global upper bound is calculated with two aims: firstly, to use it in Vertex Cover separation, and secondly, to compute the optimality gap when the algorithm finishes due to time limitations.

The global upper bound, UB^G of OP, is obtained using the dual solution π^* of the solution (y^*, x^*) of the LP_0:

$$UB^G = \sum_{i=1}^{c} \pi_i^* b_i + rc_1^* + \sum_{\substack{v \in V - \{1\} \\ rc_v^* > 0}} rc_v^* + \sum_{\substack{e \in E \\ rc_e^* > 0}} rc_e^* \tag{4.22}$$

where the reduced costs rc_v^* and rc_e^* are calculated using the dual variables π^* and c is the number of constraints.

The upper bound of a branch node, UB^N, can be calculated by subtracting the contributions of the branched edges to UB^G. Let $B_0, B_1 \subset E$ be the subset of edges branched to 0 and 1, respectively. Then, we obtain UB^N by:

$$UB^N = \sum_{i=1}^{c} \pi_i^* b_i + rc_1^* + \sum_{\substack{v \in V - \{1\} \\ rc_v^* > 0}} rc_v^* + \sum_{\substack{e \in E \\ rc_e^* > 0}} rc_e^* - \sum_{\substack{e \in B_0 \\ rc_e^* > 0}} rc_e^* + \sum_{\substack{e \in B_1 \\ rc_e^* < 0}} rc_e^* \tag{4.23}$$

4.9 Computational results

In this section, we present the results of the computational experiments. Firstly, we evaluate the new designed components for the revisited B&C algorithm (RB&C); and secondly, we compare the performance of RB&C with state-of-the-art B&C and heuristic algorithms. The software used for the experiments is publicly available on https://github.com/gkobeaga/op-solver.

The experiments are carried out using well-known instances in the literature. These instances, which are based on the TSPLIB library, were first proposed in Fischetti et al. [1998] and then extended to larger problems in Kobeaga et al. [2018]. The instances are split into two groups: medium-sized instances (up to 400 nodes) and large-sized instances (up to 7397 nodes). In total, we consider 258 benchmark instances. They are also classified into three generations (Gen1, Gen2 and Gen3) according to the definition of scores, see Fischetti et al. [1998]. For all of these three generations, the distance limitation is set as half of the TSP solution value.

In order to measure the performance of the algorithms, we compare the quality of the returned best solutions (LB) and the mean running time (in seconds) of the algorithms. In addition, in the case of the B&C algorithms, we also compare the obtained upper bounds (UB). All the experiments for the compared algorithms have been carried out using a 5-hour time limit.

In Table 4.1, we detail the values of the common parameters for all the simulations of the RB&C algorithm. They were chosen inspired by the parameters used in Applegate et al. [2007] and our preliminary experiments for the OP.

4.9.1 Evaluation of Components

In this section, we evaluate the designed components for the RB&C algorithm. We have carried out experiments with several alternative configurations of the components. To that aim, a subset of 15 OP instances were selected: 5 TSP instances (pr76, att532, vm1084, rl1323 and vm1748, inspired by the subset selected in Goldberg and Tsioutsiouliklis [2001]) with their respective score generations proposed in Fischetti et al. [1998]. Then, for each instance and generation, we have executed the different B&C configurations 5 times.

In order to evaluate our contributions, we have chosen a reference configuration, REFERENCE, that incorporates the components proposed in this chapter and compared it with its alternative configurations. The reference RB&C algorithm considers the following components:

- SEC/CC separation algorithm (Section 4.4.1):
    - i) SRK=S1S3: Uses shrinking rules S1 and S3.
    - ii) CC STRATS: Uses strategies to find extra violated CCs.

Table 4.1: Common parameters.

Parameter	Value	Description
ZERO	10^{-7}	Sensibility of fractional numbers
ADD_CUT_BATCH	250	Maximum number of cuts added to the LP_0 at once
ADD_MIN_VIOL	10^{-6}	Minimum violation of a cut to include it in the LP_0
SUBLOOP_IMPR	1%	Minimum improvement to repeat the subloops
ADD_SEC_PER_SET	50	Amount of SECs considered for each subset
ADD_PATH_MAX	500	Maximum cuts for Path inequalities separation
ADD_EGH_EPSILON	0.3	Epsilon value for the EGH blossom heuristic
PRICE_MAX_ADD	200	Maximum number of variables added to the LP_0
PRICE_RC_THRESH	10^{-5}	Minimum penalty of a variable to add to the LP_0
DEL_DUST_VAR	10^{-3}	Minimum y value to consider an edge as active
DEL_DUST_CUT	10^{-3}	Maximum slack value to consider a cut as active
DEL_MAX_AGE_CUT	5	Consecutive inactivity to delete a cut from the LP_0
DEL_MAX_AGE_VAR	100	Consecutive inactivity to delete an edge from the LP_0
XHEUR_GREEDY_XMIN	0.3	Use arcs larger than this value in PB primal heuristic
XHEUR_EA4OP_POP_SIZE	10	Population size for EA4OP
XHEUR_EA4OP_D2D	5	Iterations before checking feasibility in EA4OP
XHEUR_EA4OP_NPAR	3	Number of parents preselected in EA4OP

- Blossom separation algorithms (Section 4.4.2):
  i) EPH BLOSSOM: Uses Extended Padberg-Hong blossom heuristic.
  ii) EGH BLOSSOM: Uses Extended Grötschel-Holland blossom heuristic.
- Separation algorithms from the literature:
  i) CYCLE: Uses Cycle Cover inequalities.
  ii) EDGE: Uses Edge Cover inequalities.
  iii) PATH: Uses Path inequalities.
- Separation Loop strategy:
  i) SEP=THREE SUBLOOPS: Uses the separation loop strategy presented in Section 4.6.
- Primal heuristics (Section 4.7):
  i) XHEUR=VP + EA4OP: Constructs a small population using VP heuristic and evolves it with EA4OP.
  ii) SEP XHEUR=PB: Constructs a single solution using PB in the separation

loop.

The alternative configurations are obtained by modifying a single component in REFERENCE, while the rest of the components remain untouched. These changes to REFERENCE are made by deleting a component(-), adding a new component(+) or replacing a component (COMP=). The tested alternative strategies are the following:

- SEC/CC separation algorithm:
  i) -SRK: Does not use any shrinking technique. As a consequence, CC STRATS are not used either.
  ii) SRK=C1C2S3: The shrinking rule S1 is replaced with the rules C1C2.
  iii) -CC STRATS: Does not use strategies to find extra violated CCs.
- Blossom separation algorithms:
  i) -EPH BLOSSOM: Does not use the Extended Padberg-Hong blossom heuristic
  ii) -EGH BLOSSOM: Does not use the Extended Grötschel-Holland blossom heuristic
  iii) +FST BLOSSOM: Uses the blossom separation heuristic in Fischetti et al. [1998]
- Separation algorithms from the literature:
  i) -CYCLE COVER: Does not use Cycle Cover inequalities
  ii) -EDGE COVER: Does not use Edge Cover inequalities
  iii) +VERTEX COVER: Uses Vertex Cover inequalities
  iv) -PATH: Does not use Path inequalities
- Separation Loop strategy:
  i) SEP=TWO SUBLOOPS: The separations algorithms in the outer subloop are appended to the middle subloop.
- Primal heuristic in the branch node:
  i) XHEUR=PB: Constructs a single solution using PB heuristic.
  ii) XHEUR=VP - EA4OP: Constructs a single solution using VP heuristic.

In Table 4.2 we summarize the mean relative difference to the best achieved LB and UB, as well as the mean relative difference to the best performing configuration in terms of running time. The results grouped by instances are presented in Appendix B.3.1.

The results show that the alternatives decrease the performance of the REFERENCE configuration for the RB&C algorithm either in terms of solution quality, upper bound value, or running time. The experiments restate the importance of the shrinking techniques for the SEC/CC separation algorithm, as can be seen in the results for -SRK.

Table 4.2: Results of the alternative configurations for RB&C. In bold, the values of the alternatives that are worse than those obtained by the REFERENCE configuration.

	Gap								
	Gen1			Gen2			Gen3		
Strategy	LB	UB	Time	LB	UB	Time	LB	UB	Time
REFERENCE	0.05	0.00	262.06	0.05	0.04	23.11	0.02	0.01	44.02
- SRK	**0.13**	0.00	**532.37**	**0.10**	0.04	**25.86**	0.02	**0.02**	**134.74**
SRK=C1C2S3	0.02	0.00	88.32	**0.09**	0.04	**31.72**	0.01	0.01	**79.81**
- CC STRATS	0.02	0.00	115.91	0.04	0.01	21.85	0.01	0.01	**449.90**
- EPH BLOSSOM	**0.09**	**0.15**	208.65	**0.12**	**0.15**	**33.64**	**0.10**	**0.22**	**199.79**
- EGH BLOSSOM	0.02	0.00	**296.71**	0.04	0.04	**26.18**	**0.03**	0.01	**91.83**
+ FST BLOSSOM	0.00	0.00	**345.32**	0.04	0.00	**26.43**	**0.04**	0.00	**66.54**
- EDGE COVER	**0.11**	0.00	137.73	**0.13**	0.04	**30.04**	**0.05**	0.01	35.50
- CYCLE COVER	**0.06**	0.00	124.79	0.02	0.04	**25.60**	**0.03**	0.01	**48.18**
- PATH	**0.08**	0.00	183.86	**0.10**	0.04	**32.00**	**0.03**	0.01	**69.01**
+ VERTEX COVER	0.05	0.00	61.10	0.03	0.04	22.33	**0.03**	0.01	**104.82**
SEP: TWO SUBLOOPS	0.05	0.00	**315.34**	**0.06**	0.04	17.05	**0.03**	0.01	**164.44**
XHEUR=PB	**0.08**	0.00	179.14	**0.12**	0.01	2.37	**0.04**	0.01	62.74
XHEUR=VP - EA4OP	0.02	0.00	222.46	**0.07**	0.04	7.17	0.01	0.01	**168.63**

It is not only worse not using the shrinking in terms of time, but indeed, the obtained LB values are also worse. In addition, the results suggest that the S1 shrinking technique, which is considered in REFERENCE, might be preferable to the C1C2 technique. Regarding the CC STRATS, the results for Gen3 suggest that not considering the strategies to find extra violated CCs might have a negative impact on the running time of the algorithm.

Next, looking at the separation algorithms for blossoms, the results show that the EPH heuristic is crucial in the RB&C, particularly, if we focus on the obtained LB and UB values. From the table, we can also extract that the EGH heuristic improves the running time of the B&C algorithm. Alternatively, although the FST blossom heuristic might improve the quality of the solutions, it reports worse running times.

With respect to the rest of the separation algorithms proposed in the literature for the OP, we include in REFERECE all but Vertex Cover inequalities. This way, the RB&C uses the same families of cuts as in Fischetti et al. [1998], which enables us to evaluate the contributions in this chapter in a better way.

Finally, the experiments show that the VP primal heuristic plays an important role in obtaining better LB values, particularly for large problems, as can be seen in the detailed results in Appendix B.3.1. However, solving the VP primal heuristic in the branch node is more costly than PB primal heuristic, hence the running time of the

RB&C is worsened in the smallest instances. Similarly, by using the EA4OP to improve the results by VP heuristic, the obtained LB values are improved in large problems at the expense of worsening the running time in the smallest instances.

4.9.2 Comparison with state-of-the-art Algorithms

The proposed reference RB&C has been compared with the state-of-the-art B&C algorithm in Fischetti et al. [1998] (FST) and two state-of-the-art heuristics, Kobeaga et al. [2018] (EA4OP) and Santini [2019] (ALNS). The detailed results can be found in Appendix B.3.2.

Three notes before moving on to the discussion. First, the FST code reports the running times using one trailing digit while the rest of the algorithms report the times using two trailing digits. In order to make use of the reported times in the literature of the FST, we round the obtained times by the RB&C to one trailing digit when we compare it with the FST algorithm. Secondly, the FST returns a false optimum for pa561 in Gen1. We assume that this is a consequence of the rounding sensibility and we accept as valid the rest of the reported optima by FST. Thirdly, eight instances (rat99, rat195, tsp225, pa561, rat575, rat783, nrw1379, and fnl4461) of Gen3 have been excluded for the comparison of the RB&C with the EA4OP and the ALNS, due to an issue in the generation of scores of the instances used by those algorithms. Since the results of the current comparison are clear enough, we have discarded rerunning the experiments with the updated scores.

First, we compare the RB&C algorithm with the B&C by Fischetti et al. [1998]. The results of the FST algorithm were updated using CPLEX12.5 in Kobeaga et al. [2018], which is the same version of CPLEX used for the experiments of RB&C. Moreover, the new experiments are run on the same machine with the same amount of reserved memory (4GB). In Table 4.3 we summarize, by size and generation, the number of instances returning a feasible solution, #, the obtained optimality certifications, OPT, the number of best-known solution (LB), and upper bound (UB) values.

In Table 4.3 it can be seen that the RB&C algorithm is able to obtain the best-known solutions value in all the medium-sized instances.

Comparison with FST algorithm

Moving on to large-sized instances, the superiority of the RB&C algorithm compared to the FST approach becomes evident. While the FST algorithm fails to output a solution in almost half of the instances (mainly because of running out of memory), the RB&C algorithm returns a solution for every instance. Moreover, it obtains the best-known solution in significantly more instances than FST (245 against 170) and UB (249 against 173) values. Even more, it obtains more optimality certifications (180 against 165).

Table 4.3: Comparison of the number of instances in which a feasible solution (#), an optimal (OPT), a best-known solution (LB) or a best upper bound value (UB) were obtained.

		#		OPT		LB		UB	
Size	Gen	FST	RB&C	FST	RB&C	FST	RB&C	FST	RB&C
Medium	Gen1	45	45	**45**	44	45	45	**45**	44
	Gen2	45	45	45	45	45	45	45	45
	Gen3	45	45	45	45	45	45	45	45
Large	Gen1	21	**41**	12	**24**	13	**39**	13	**40**
	Gen2	22	**41**	9	**10**	9	**36**	13	**38**
	Gen3	29	**41**	9	**12**	13	**35**	12	**37**
	All	207	**258**	165	**180**	170	**245**	173	**249**

Table 4.4: Comparison of the number of obtained optimal solutions (OPT), number of best-known solutions (LB) and number of best upper bounds (UB) in the instances that FST does return a solution.

		OPT		LB		UB		Time	
	#	FST	RB&C	FST	RB&C	FST	RB&C	FST	RB&C
Gen1	66	1	**4**	0	**6**	2	**8**	15	**40**
Gen2	67	**1**	0	0	**11**	3	**9**	25	**27**
Gen3	74	1	**3**	1	**14**	4	**17**	23	**33**
All	207	3	**7**	1	**31**	9	**34**	63	**100**

In Table 4.4 we compare the quality of the solutions and running times, restricted to those instances in which FST actually returns a solution. We particularly focus on the number of solutions (optimality certifications, best-known solutions and upper bounds) that are new in the literature, i.e., values not obtained by the rest of the algorithms. Thus, for the lower-bound values, we also take into account the results obtained by the EA4OP and ALNS heuristics. Additionally, we show the number of instances in which the considered B&C algorithms are faster than the competitor. When we restrict the considered instances to the instances where the FST obtains a feasible solution, the RB&C outperforms the results of the FST. While the FST obtains 1 new best-known solution (not obtained by any other algorithm) and 9 new UB values, the RB&C obtains 31 LB and 34 UB new values. In the same set of instances, the FST obtains 3 optimality certifications that the RB&C is not able to obtain, while the RB&C obtains 7 optimality certifications that the FST is unable to obtain. Moreover, it turns out that the RB&C

is faster than the FST in 100 instances while the FST is faster than the RB&C in 63 instances.

Next, we compare the RB&C algorithm against state-of-the-art algorithms in terms of solution quality, running time, and Pareto efficiency. In Table 4.5 and Table 4.6 the algorithms are compared pairwise and instance-by-instance for medium-sized and large-sized instances respectively. The aim is to measure the number of instances where an algorithm is simultaneously as least as fast as the opponent and obtains a better quality solution.

Table 4.5: Comparison in medium-sized instances against state-of-the-art algorithms in terms of quality, time and Pareto efficiency.

	Gen1			Gen2			Gen3		
	EA4OP	tie	RB&C	EA4OP	tie	RB&C	EA4OP	tie	RB&C
Quality	0	30	**15**	0	14	**31**	0	15	**27**
Time	15	0	**30**	**37**	0	8	**39**	0	3
Pareto	7	0	**30**	**10**	0	8	**13**	0	3
	ALNS	tie	RB&C	ALNS	tie	RB&C	ALNS	tie	RB&C
Quality	0	40	**5**	0	29	**16**	0	29	**13**
Time	1	0	**44**	4	0	**41**	8	0	**34**
Pareto	1	0	**44**	1	0	**41**	5	0	**34**
	FST	tie	RB&C	FST	tie	RB&C	FST	tie	RB&C
Quality	0	45	0	0	45	0	0	45	0
Time	14	6	**25**	17	2	**26**	18	1	**26**
Pareto	14	6	**25**	17	2	**26**	18	1	**26**

Comparison in medium-sized instances

Table 4.5 shows that the RB&C algorithm is competitive in medium-sized instances. Compared to the ALNS heuristic and FST algorithm, it obtains better Pareto efficiency results in the three generations. Comparing it to EA4OP, the Pareto efficiency is lower because the heuristic is a faster algorithm. Nevertheless, the RB&C obtains much better quality solutions.

Table 4.6: Comparison in large-sized instances against state-of-the-art algorithms in terms of quality, time and Pareto efficiency.

	Gen1			Gen2			Gen3		
	EA4OP	tie	RB&C	EA4OP	tie	RB&C	EA4OP	tie	RB&C
Quality	1	0	**40**	5	0	**36**	3	0	**33**
Time	**39**	0	2	**40**	1	0	**35**	1	0
Pareto	1	0	**2**	**5**	0	1	**3**	0	1
	ALNS	tie	RB&C	ALNS	tie	RB&C	ALNS	tie	RB&C
Quality	2	2	**37**	4	1	**36**	4	0	**32**
Time	6	11	**24**	**13**	25	3	**13**	19	4
Pareto	4	0	**34**	5	0	**24**	4	0	**20**
	FST	tie	RB&C	FST	tie	RB&C	FST	tie	RB&C
Quality	0	13	**28**	0	9	**32**	3	11	**27**
Time	1	5	**35**	8	13	**20**	5	17	**19**
Pareto	1	1	**39**	8	0	**33**	7	2	**32**

Comparison in large-sized instances

Table 4.6 shows that RB&C is the best performing algorithm in large-sized instances. Particularly, it behaves better than the FST algorithm, obtaining the best quality and time solutions in most of the instances, hence obtaining better Pareto results. The ALNS algorithm is able to return some solutions with better quality or running time, however, overall, the RB&C performs better in large-sized instances. The EA4OP metaheuristic is faster than the B&C but, in general, obtains worse quality solutions.

Finally, in Table 4.7, we summarize the new best-known results obtained in the experiments. The RB&C algorithm obtains 18 new optimality certifications, 76 new best-known solution values and 85 new upper-bound values.

4.10 Conclusions

We have presented a revisited version of the B&C algorithm for the OP that brings multiple contributions together. We have proposed a joint separation algorithm for SECs

Table 4.7: New best-known optimum, lower bound and upper bound values.

	OPT	LB	UB
Gen1	12	25	28
Gen2	2	27	28
Gen3	4	24	29
All	18	76	85

and CCs, which efficiently uses the shrinking technique for cycle problems by reducing the adverse effects of the shrinking for CCs. We have developed two blossom heuristics for cycle problems which generalize the well-known approaches in the literature of the TSP. We have designed an efficient variable pricing procedure for the OP which enables us to avoid repetitive calculations and to skip the exact calculation of the reduced cost of some variables. We have proposed a separation loop for the OP that takes into consideration the different contributions and separation costs of the valid inequalities. We have used alternative primal heuristics, one of which is based on a metaheuristic, and a mechanism to update the global upper bound during the branching phase to tighten the lower and upper bounds for the cases when the algorithm finishes without an optimality certification.

The experiments have shown that the RB&C algorithm for OP is a more efficient approach than the state-of-the-art B&C algorithm. The introduced algorithm has increased the number of solved problems, obtained better running times in more instances, succeeded in returning new optimality certifications, new best known solutions, and new upper-bound values for large problems. Additionally, it has been shown that the RB&C algorithm obtains better quality solutions than the state-of-the-art heuristics for the OP within the 5-hour running time limit.

Nevertheless, there are many research lines that remain open after this work. One of the most demanding aspects to improve in the presented approach is the implementation of advanced branching techniques. The use of more general cuts, such as combs and clique trees, and the development of their respective separation algorithms for cycle problems might help to improve the performance of the RB&C algorithm. All these future contributions might help to solve the remaining instances until optimality, but we can anticipate it will not be an easy challenge. Implementing the contributions in this chapter to other cycle problems which are different from the OP will definitely help to comprehend their importance in the context of cycle problems with a more general view.

CHAPTER 5

Software for OP

The implementation of the proposed algorithms has been an important part of this thesis. Although we have released a repository of software for each chapter, all the algorithms implemented during the thesis have been included in our last software repositoy for the B&C algorithm. This repository, which is publicly available under the Apache 2.0 license at `https://github.com/gkobeaga/op-solver`, is an extensive work written in C. In table 5.1 a summary of the repository contents can be seen.

Table 5.1: Summary of the repository contents.

Language	Files	Lines	Code	Blanks
Bash	1	7	7	0
Automake	28	414	321	85
C Header	25	2654	2269	385
C	139	26972	23649	3317
Total	193	30047	26246	3787

A large part of the source code related with the B&C algorithm has been inspired by the Concorde solver developed in Applegate et al. [2007] for the TSP, which is publicly available at `http://www.math.uwaterloo.ca/tsp/concorde.html`. We have also used the implementation of the B&C proposed in Fischetti et al. [1998], which had been provided by the authors, as a reference for some of the separation algorithms.

When implementing the algorithms for the OP, particularly the B&C algorithm, the are several subproblems (Minimum Cut Problem/Maximum Flow Problem, Minimum Spanning Tree Problem, Knapsack Problem, Linear Problem, Integer Problem, Cycle Problem) and data-strucures (Graph, Spatial Data) involved. The repository is structured as follows:

- Data-sctructures:
  (a) Graph [graph]
    i. Constructors [graph/graph.c]
    ii. Hash [graph/arc_hash.c]
    iii. Connected components [graph/connect.c]
    iv. Minimum-cut problem [graph/{maxflow.c, mincut.c}]
    v. Gomory-hu trees [graph/ghtree.c]
    vi. Minimum Spanning Tree [graph/mst.c]
  (b) Data [data/]
    i. Read distance [data/io/]
    ii. Nearest Neighbor (k-NN) [data/nearest/]
      i. k-d trees (Concorde) [data/nearest/kdtree]
- Problems:
  (a) Knapsack Problem [prob/kp]
    i. Initialization
    ii. Exact [prob/kp/exact]
      i. Branch-and-Bound [prob/kp/exact/bab.c]
  (b) Linear Problem [prob/lp]
    i. External LP sover: `IBM ILOG CPLEX` [prob/lp/lib/cplex.c]
  (c) Integer Problem [prob/ip]
    i. Dependency: LP
    ii. Exact [prob/ip/exact]
      i. Branch-and-Bound [prob/ip/exact/bac]
  (d) Travelling Salesperson Problem [prob/tsp]
    i. Dependency: CP
    ii. Initialization [prob/tsp/init]
    iii. Heuristic [prob/tsp/heur]
      i. k-opt [prob/tsp/heur/kopt]
      ii. Lin-Kernighan (Concorde) [prob/tsp/heur/linkern]
  (e) Cycle Problem [prob/cp]
    i. Dependency: LP, IP, KP, TSP

ii. Initialization [prob/cp/init]

iii. Exact [prob/cp/exact]

  i. Branch-and-Cut [prob/cp/exact/bac]

iv. Heuristic [prob/cp/heur]

  i. Evolutionary Algorithm [prob/cp/heur/ea]

• Each problem (and algorithm) has an associated enviroment where the specific enviromental variables (i.e. verbosity), parameters and statistics are stored:

Problem Enviroment [prob/*/env.c]

i. Parameters [prob/*/param.c]

ii. Statistics [prob/*/stats.c]

iii. Initialization Enviroment [prob/*/init/env.c]

  i. Parameters [prob/*/init/param.c]

  ii. Statistics [prob/*/init/stats.c]

iv. Heuristic Enviroment [prob/*/heur/env.c]

  i. Parameters [prob/*/heur/param.c]

  ii. Statistics [prob/*/heur/stats.c]

v. Exact Enviroment [prob/*/exact/env.c]

  i. Parameters [prob/*/exact/param.c]

  ii. Statistics [prob/*/exact/stats.c]

5.1 Installation

The software is build using the GNU Autools suite. First, download the source code,

Listing 5.1: Clone the repository

```
git clone https://github.com/gkobeaga/op-solver
cd op-solver
```

install the dependencies,

Listing 5.2: Install the dependencies

```
sudo apt install autoconf automake libtool m4 libgmp-dev
```

and generate the `configure` script.

Listing 5.3: Generate the `configure` script

```
./autogen.sh
mkdir -p build && cd build
```

Since the external LP solver is proprietary software, there are two options to install our software: to build only the heuristic algorithm or to build both the heuristic and the exact algorithms.

5.1.1 Install Heuristic Algorithm

By default, the solver is built only with the heuristic algorithm:

Listing 5.4: Build only the heuristic

```
make clean
../configure
make
```

5.1.2 Install Heuristic and Exact Algorithms

To build the exact algorithm, you need to have the `IBM ILOG CPLEX` installed in your system. To build the ‘op-solver‘ with the exact algorithm:

Listing 5.5: Build the heuristic and the B&C algorithm

```
make clean
../configure --with-cplex=<CPLEX_PATH>
make
```

For instance, if CPLEX is installed in /opt/ibm/ILOG/CPLEX_Studio125/cplex/ the configuration is carried out as follows:

Listing 5.6: Example of the configuration with cplex

```
../configure --with-cplex=/opt/ibm/ILOG/CPLEX_Studio125/cplex/
```

5.2 Usage

In order to use the OP solvers, download first the benchmark instances for the OP:

Listing 5.7: Download the OP instances

```
cd build
git clone https://github.com/bcamath-ds/OPLib.git
```

To solve the problem using the EA4OP algorithm:

Listing 5.8: Solve OP with EA4OP

```
./src/op-solver opt --op-exact 0 OPLib/instances/gen3/kroA150-gen3
    -50.oplib
```

To solve the OP using the revisited Branch-and-Cut algorithm(RB&C):

Listing 5.9: Solve OP with RB&C

```
./src/op-solver opt --op-exact 1 OPLib/instances/gen3/kroA150-gen3
    -50.oplib
```

You can increase the verbosity of the RB&C with:

Listing 5.10: Increase verbosity

```
./src/op-solver opt --op-exact 1 --op-exact-bac-verbose 1 OPLib/
    instances/gen3/kroA150-gen3-50.oplib
```

When the B&C algorithm finishes, it writes the solution in the "solution" directory:

Listing 5.11: Solution file for kroA150-Gen3

```
NAME : kroA150
TYPE : OP
DIMENSION : 150
COST_LIMIT : 13262.00
ROUTE_NODES : 79
ROUTE_SCORE : 5039.00
ROUTE_COST : 13246.00
NODE_SEQUENCE_SECTION
1
93
28
58
61
25
81
69
64
40
54
2
144
114
44
50
```

```
116
82
126
95
13
76
33
146
103
37
5
52
78
96
39
101
121
30
107
112
132
29
46
3
14
48
100
71
41
136
128
43
123
115
120
149
55
83
34
135
140
125
51
87
145
9
117
7
57
```

```
20
12
27
86
150
62
60
77
110
23
98
91
109
47
-1
DEPOT_SECTION
1
-1
EOF
```

It also writes the execution statistics in the “bac-stats.json” file in the following format:

Listing 5.12: Statistics for kroA150-Gen3


```
{
  "prob": {
    "name": "kroA150",
    "n": 150,
    "d0": 13262
  },
  "sol": {
    "val": 5039,
    "cap": 13246,
    "sol_ns": 79,
    "lb": 5039,
    "ub": 5039
  },
  "param": {
    "sep_logical": 1,
    "sep_sec_comps": 1,
    "sep_sec_exact": 3,
    "sep_sec_cc_2": 0,
    "sep_sec_cc_extra": 1,
    "sep_blossom_fst": 0,
    "sep_blossom_eph": 1,
    "sep_blossom_egh": 1,
```

```
    "sep_cover_edge": 1,
    "sep_cover_vertex": 0,
    "sep_cover_cycle": 1,
    "sep_path": 1,
    "sep_loop": 1,
    "sep_srk_rule": 4,
    "sep_srk_s2": 0,
    "sep_srk_s3": 1,
    "sep_srk_extra": 1,
    "xheur_vph": 1,
    "xheur_vph_meta": 1
  },
  "stats": {
    "time": 34427,
    "active_sep_logical": 2207,
    "success_sep_logical": 265,
    "total_sep_logical": 460,
    "active_sep_sec_comps": 2207,
    "success_sep_sec_comps": 31,
    "total_sep_sec_comps": 664,
    "time_sep_sec_comps": 151,
    "active_sep_sec_exact": 891,
    "success_sep_sec_exact": 726,
    "total_sep_sec_exact": 6072,
    "time_sep_sec_exact": 3171,
    "active_sep_blossom_fast": 914,
    "success_sep_blossom_fast": 127,
    "total_sep_blossom_fast": 274,
    "time_sep_blossom_fast": 315,
    "active_sep_blossom_ghfast": 897,
    "success_sep_blossom_ghfast": 94,
    "total_sep_blossom_ghfast": 149,
    "active_sep_blossom_mst": 0,
    "success_sep_blossom_mst": 0,
    "total_sep_blossom_mst": 0,
    "time_sep_blossom_mst": 0,
    "active_sep_cover_edge": 486,
    "success_sep_cover_edge": 48,
    "total_sep_cover_edge": 48,
    "time_sep_cover_edge": 941,
    "active_sep_cover_cycle": 851,
    "success_sep_cover_cycle": 2,
    "total_sep_cover_cycle": 2,
```

```
    "time_sep_cover_cycle": 52,
    "active_sep_cover_vertex": 0,
    "success_sep_cover_vertex": 0,
    "total_sep_cover_vertex": 48,
    "time_sep_cover_vertex": 0,
    "active_sep_path": 482,
    "success_sep_path": 22,
    "total_sep_path": 111,
    "time_sep_path": 196,
    "time_sep_sep_loop": 12164,
    "time_sep_sep_loop_it": 0,
    "time_sep_sep_loop_inner": 2830,
    "time_sep_sep_loop_inner_it": 1015,
    "time_sep_sep_loop_middle": 10833,
    "time_sep_sep_loop_middle_it": 9368,
    "time_sep_sep_loop_outer": 12164,
    "time_sep_sep_loop_outer_it": 2450,
    "time_age_cut": 687,
    "time_age_vars": 907,
    "time_add_vars": 6216,
    "time_add_cuts": 4957,
    "time_xheur_branch": 30,
    "time_xheur_sep": 2330
  },
  "timestamp": 1606759120605,
  "event": "stats_summary",
  "env": "cp_exact_bac",
  "seed": 127591,
  "pid": 148865
}
```

CHAPTER 6

Conclusions, Future Work and Contributions

6.1 Conclusions

In this thesis, we have developed two algorithms to solve large-scale orienteering problems: a heuristic evolutionary algorithm and an exact Branch-and-Cut algorithm. As part of the research carried out for the exact algorithm, we have extended, from the literature of the TSP, the support graph shrinking tecniques for cycle problems.

In Chapter 2 of this work, we have presented an efficient evolutionary algorithm for the OP. Essentially, the algorithm follows the steady-state genetic algorithm schema. The proposed method maintains unfeasible solutions during the search and considers specific operators to recover it when required. It allows us to obtain high quality solutions without being penalized in terms of computational time. Furthermore, the parameter $d2d$ helps to strike a balance between solution quality and computational time.

The Edge Recombination crossover, originally proposed for the TSP, has been adapted for the OP. We consider this adaptation of the Edge Recombination crossover as a contribution to the solution of cycle problems in general. In addition to the problems that consist of permutations, this adaptation also allows us to deal with a wider range of problems whose solution space consists of simple cycles.

Another contribution that we find remarkable for routing problems is the proposed add operator. When the distance matrix is given by spatial points, its design allows the use of a data structure, i.e., k-d tree, that strongly reduces the computational cost, improving the overall results.

The computational experiments have shown that several characteristics are essential for the effectiveness of the EA4OP, the use of unfeasible solutions during the search process being the most relevant feature. All in all, the EA4OP proves to be an efficient algorithm for the OP. In comparison to the state-of-the-art algorithms, not only does the EA4OP obtain competitive results in medium-sized instances, but it also achieves outstanding results in large-sized instances in terms of quality with low execution times.

In Chapter 3 of this dissertation we have successfully generalized, for cycle problems, the global (C1, C2 and C3) and SEC specific (S1, S2 and S3) shrinking rules proposed in the literature of the TSP. The obtained computational results for the shrinking in the OP are remarkable. The results clearly show that the shrinking technique considerably improves the running time of the separation algorithms for SECs.

Part of the chapter focuses on exact SEC separation algorithms for cycle problems. We have extended from the TSP two exact algorithms (Algorithm DH and Algorithm EPG). The proposed separation algorithms were shown to be more efficient in the OP than the exact algorithm used so far in the literature (the adaptation of the classical Hong's approach). The importance of the detailed extension of the Padberg-Grötschel approach, Algorithm EPG, lies in the fact that in cycle problems, in general, the global minimum cut of a support graph might not generate a violated SEC, while violated SECs in the same graph exist.

In Chapter 4 a revisited version of the B&C algorithm for the OP that brings multiple contributions together is presented. We have proposed a joint separation algorithm for SECs and CCs, which efficiently uses the shrinking technique for cycle problems by reducing the adverse effects of the shrinking for CCs. Two blossom heuristics for cycle problems which generalize the well-known approaches in the literature of the TSP have been developed. We have designed an efficient variable pricing procedure for the OP which enables us to avoid repetitive computations and to skip the exact calculation of the reduced cost of some variables. A separation loop for the OP has been proposed which takes into consideration the different contributions and separation costs of the valid inequalities. Alternative primal heuristics are used, one of which is based on a metaheuristic, and a mechanism to update the global upper bound during the branching phase to tighten the lower and upper bounds for the cases when the algorithm finishes without an optimality certification.

The experiments have shown that the RB&C algorithm for OP is a much more efficient approach than the state-of-the-art B&C algorithm. The introduced algorithm has increased the number of solved problems, obtained better running times in more instances, succeeded in returning new optimality certifications, new best-known solutions, and new upper-bound values for large problems. Additionally, it has been shown that the RB&C algorithm obtains better quality solutions than the state-of-the-art heuristics for the OP within the 5-hour running time limit.

Finally, in Chapter 5 we show how to install and use the software developed during the thesis period.

In conclusion, we have proposed two algorithms in this thesis that are state-of-the-art in their respective fields for the OP, especially for instances with a large number of nodes. Depending on the goal, one can take advantage of the exact RB&C algorithm or the heuristic EA4OP algorithm. The proposed exact algorithm was shown to be the most appropriate when the available computational time to obtain a solution is high since it obtains the best quality solutions and returns the best upper bound of the problems.

Conversely, if a quick solution is required, the EA4OP was shown to be the fastest in large-sized instances, while still providing acceptable solution quality.

6.2 Future Work

Although the results of the EA4OP and the RB&C are outstanding, there are some aspects of these algorithms that could be improved:

(1) *Improve the solution initialization*

In Chapter 2 we have proposed a strategy to select a subset of vertices to include in the initial solutions, where all the nodes were sampled using the same probability parameter for the Bernoulli distribution. However, giving a different a priori probability to each node might contribute to obtain better initial solutions. These probabilities might depend on the score of the node, the distance from the depot, or the number of nodes in the neighborhood.

(2) *Study the k-d tree based node insertion local search with more detail.*

We have extensively used the k-d tree based local search in both algorithms. This choice was made based on the preliminary experiments carried out in a subset of instances of the OP, where a considerable speedup was seen. We believe that the k-d tree based node insertion is a remarkable contribution, not only for the OP but, for problems where only a subset of vertices might be visited. A detailed comparison, in multiple related problems, against commonly used node insertion procedures in the literature would help to understand the real contribution of the new local search approach.

(3) *Apply the shrinking either for other valid cycle inequalities of the OP or for other cycle problems.*

In Chapter 3 we studied the contribution of the shrinking in accelerating the SEC separation for OP. However, three of the presented rules (C1, C2 and C3) are safe for all the valid inequalities. It would be interesting to analyze if the shrinking preprocess is also useful to speedup the separation of other valid inequalities for OP. Another possibility to extend the work is to evaluate the shrinking technique in the separation problems for other cycle problems.

(4) *Use advanced branching techniques in the RB&C.*

The branching in the RB&C has been carried out in a classical way following a depth-first-search, where the edges are branched first to 1 and then to 0. In order to select the edge variable to branch, we used the classical branching strategy: the edge e, with the fractional value closest to 0.5 is selected. This is the simplest possible branching strategy and more sophisticated alternatives should be studied.

(5) *Use more general valid cuts in the RB&C.*

The use of more general cuts, such as combs and clique trees, and the development of their respective separation algorithms for cycle problems might help to improve the performance of the RB&C algorithm.

(6) *Parallelize the EA4OP and RB&C.*

Both proposed algorithms in the thesis were implemented sequentially and it would be interesting to study the parallelization of these algorithms. Regarding the EA4OP, the solutions initialization, the local search and feasibility recover procedures are easily parallelizable. The parallelization of the RB&C is more complicated, but an effort in this direction could provide new results.

(7) *Apply the EA4OP and RB&C to variants of the OP.*

Another research line of particular interest is the application of the EA4OP and the RB&C to solve variants of OP presented in Chapter 1.

(8) *Improve the availability and usability of the software*

The software have been developed with its extension to cycle problems and OP variants in mind. We plan to improve its availability to allow other researchers and developers to use it.

6.3 Contributions

Publications:

- [Kobeaga et al., 2018] *An Efficient Evolutionary Algorithm for the Orienteering Problem*, G. Kobeaga, M. Merino, and J.A Lozano. In Computers & Operations Research, volume 90, pages 42–59.
- [arXiv:2004.14574]. *On solving Cycle Problems with Branch-and-Cut: Extending Shrinking and Exact Subcycle Elimination Separation Algorithms*, G. Kobeaga, M. Merino, and J.A Lozano. Submitted to Annals of Operations Research.
- [arXiv:2011.02743]. *A revisited branch-and-cut algorithm for large-scale orienteering problems*, G. Kobeaga, M. Merino, and J.A Lozano. Submitted to Computers and Operations Research.

Presentations in International Conferences and Summer Schools:

- *A revisited branch-and-cut algorithm for the orienteering problem*, G. Kobeaga, M. Merino, and J.A Lozano, 30th European Conference on Operational Research EURO in Dublin (Ireland), June 2019.
- *An evolutionary algorithm to solve large orienteering problems efficiently*, G. Kobeaga, M. Merino, and J.A Lozano, Metaheuristics Summer School MESS in Acireale-Catania (Italy), July 2018.
- *Adapting efficient TSP exact algorithms for large orienteering problems*, G. Kobeaga, M. Merino, and J.A Lozano, ECCO XXXI Conference in Fribourg (Switzerland),

June 2018.

- *Solving large-sized orienteering problem instances using an evolutionary algorithm*, G. Kobeaga, M. Merino, and J.A Lozano, Joint EURO/ORSC/ECCO Conference in Koper (Slovenia), May 2017.

Diffusion activities:

- *Algorithms for large orienteering problems*, G. Kobeaga, M. Merino, and J.A Lozano, Operational Research Group in Brescia (Italy), October 2019.
- *Orientazio Problemak*. G. Kobeaga. Zientzialari 98 irratsaioa (https://www.bilbohiria.eus/56107). Bilbao Hiria Irratia (Radio Bilbao Hiria), 2019/04/25.
- *Orientazio problema handiak ebazteko algoritmo zehatzak arintzen*, G. Kobeaga, M. Merino, and J.A Lozano, Matematikari Euskaldunen III. Topaketak in UEU Eibar July 2018.
- *Orientazio Problema*. G. Kobeaga. Euskal Herriko Unibertsitatearen (UPV/EHU) Kultura Zientifikoko Katedra (https://vimeo.com/266644370). 2018/04/26.
- *On solving the Orienteering Problem via an efficient Evolutionary Algorithm*. G. Kobeaga, M. Merino, J. A. Lozano. In 6as Jornadas de Investigación de la Facultad de Ciencia y Tecnología, in UPV/EHU Leioa, 2018.
- *Integer programming for combinatorial optimization problems*, G. Kobeaga, M. Merino, and J.A Lozano, Intelligent Systems Group Seminar in UPV/EHU Donostia, March 2018.
- *Un algoritmo evolutivo eficiente para el Problema de Orientación*, G. Kobeaga, M. Merino, and J.A Lozano, Grupo de Optimización Estocástica, UPV/EHU Leioa, December 2017.
- *Ordering nodes for insertion procedures in the Orienteering Problem*, G. Kobeaga, M. Merino, and J.A Lozano, Intelligent Systems Group Seminar in UPV/EHU Donostia, May 2017.
- *Orientazio Problema*, G. Kobeaga, M. Merino, and J.A Lozano, Matematikari Euskaldunen II. Topaketak, in UEU Eibar, July 2016.
- *Orienteering Problems*, Intelligent Systems Group Seminar, UPV/EHU May 2016
- *Ba al dago matematikaririk Marten?* G. Kobeaga, M. Merino. Blog de la cátedra de Cultura Científica de la UPV/EHUko Kultura Zientifikoko Katedra, Zientzia Kaiera, 2016-02-29.

Contributed Presentations:

- *Stochastic Network Optimization for Intelligent Transport and Logistics*. U. Aldasoro, L. Aranburu, I. Eguia, L.F. Escudero, M. A. Garín, I. Gago, G. Kobeaga, M. Merino, G. Pérez, C. Pizarro and A. Unzueta. In 7as Jornadas de Investigación de la Facultad de Ciencia y Tecnología, 2020.

- *Optimization and Risk Management.* In U. Aldasoro, L. Aranburu, I. Eguia, L.F. Escudero, M. A. Garín, G. Kobeaga, M. Merino, G. Pérez, C. Pizarro and A. Unzueta. In 6as Jornadas de Investigación de la Facultad de Ciencia y Tecnología, 2018.

Software and Research Material:

- `https://github.com/bcamath-ds/compass`: implementation of the evolutionary algorithm for the OP. Software used in Kobeaga et al. [2018].
- `https://github.com/gkobeaga/cpsrksec`: implementation of the shrinking and exact SEC separation algorithms for cycle problems. Software used in Kobeaga et al. [2020a].
- `https://github.com/gkobeaga/op-solver`: implementation of the Branch-and-Cut algorithm for the OP. Software used in Kobeaga et al. [2020b].
- `https://github.com/bcamath-ds/OPLib`: A repository of the benchmark instances for the OP.

References

Angelelli, E., Archetti, C., Filippi, C., and Vindigni, M. (2017). The probabilistic orienteering problem. *Computers & Operations Research*, 81:269 – 281.

Angelelli, E., Archetti, C., and Vindigni, M. (2014a). The clustered orienteering problem. *European Journal of Operational Research*, 238(2):404 – 414.

Angelelli, E., Bazgan, C., Speranza, M. G., and Tuza, Z. (2014b). Complexity and approximation for traveling salesman problems with profits. *Theoretical Computer Science*, 531:54 – 65.

Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2007). *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA.

Archetti, C., Corberán, A., Plana, I., Sanchis, J. M., and Speranza, M. G. (2016). A branch-and-cut algorithm for the orienteering arc routing problem. *Computers & Operations Research*, 66:95 – 104.

Archetti, C., Speranza, M. G., Corberán, A., Sanchis, J. M., and Plana, I. (2014a). The team orienteering arc routing problem. *Transportation Science*, 48(3):442–457.

Archetti, C., Speranza, M. G., and Vigo, D. (2014b). Vehicle routing problems with profits. In *Vehicle Routing: Problems, methods, and applications*, chapter 10, pages 273–297. MOS-SIAM Series on Optimization.

Balas, E. (1975). Facets of the knapsack polytope. *Mathematical Programming*, 8(1):146–164.

Balas, E. (1989). The prize collecting traveling salesman problem. *Networks*, 19(6):621–636.

Bauer, P. (1997). The circuit polytope: Facets. *Mathematics of Operations Research*, 22(1):110–145.

Bauer, P., Linderoth, J., and Savelsbergh, M. (2002). A branch and cut approach to the cardinality constrained circuit problem. *Mathematical Programming*, 91:307–348.

Bentley, J. L. (1990). K-d trees for semidynamic point sets. In *Proceedings of the Sixth Annual Symposium on Computational Geometry*, SCG '90, page 187–197, New York, NY, USA. Association for Computing Machinery.

Bérubé, J.-F., Gendreau, M., and Potvin, J.-Y. (2009). A branch-and-cut algorithm for the undirected prize collecting traveling salesman problem. *Networks*, 54:56–67.

Bianchessi, N., Mansini, R., and Speranza, M. G. (2018). A branch-and-cut algorithm for the team orienteering problem. *International Transactions in Operational Research*, 25(2):627–635.

Bouly, H., Dang, D.-C., and Moukrim, A. (2010). A memetic algorithm for the team orienteering problem. *4OR-A Quarterly Journal of Operations Research*, 8(1):49–70.

Boussier, S., Feillet, D., and Gendreau, M. (2007). An exact algorithm for team orienteering problems. *4OR quarterly journal of the Belgian, French and Italian Operations Research Societies*, 5:211–230.

Campbell, A. M., Gendreau, M., and Thomas, B. W. (2011). The orienteering problem with stochastic travel and service times. *Annals of Operations Research*, 186:61–81.

Campos, V., Martí, R., Sánchez-Oro, J., and Duarte, A. (2014). Grasp with path relinking for the orienteering problem. *Journal of the Operational Research Society*, 65(12):1800–1813.

Chao, I., Golden, B. L., and Wasil, E. A. (1996a). A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475 – 489.

Chao, I.-M., Golden, B. L., and Wasil, E. A. (1996b). The team orienteering problem. *European Journal of Operational Research*, 88(3):464 – 474.

Chen, C., Cheng, S.-F., and Lau, H. (2014). Multi-agent orienteering problem with time-dependent capacity constraints. *Web Intelligence and Agent Systems*, 12:347–358.

Coullard, C. R. and Pulleyblank, W. R. (1989). On cycle cones and polyhedra. *Linear Algebra and its Applications*, 114-115:613 – 640. Special Issue Dedicated to Alan J. Hoffman.

Crowder, H. and Padberg, M. W. (1980). Solving large-scale symmetric travelling salesman problems to optimality. *Management Science*, 26(5):495–509.

Dang, D.-C., El-Hajj, R., and Moukrim, A. (2013). A branch-and-cut algorithm for solving the team orienteering problem. In Gomes, C. and Sellmann, M., editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 332–339, Berlin, Heidelberg. Springer Berlin Heidelberg.

Dell'Amico, M., Maffioli, F., and Värbrand, P. (1995). On prize-collecting tours and the asymmetric travelling salesman problem. *International Transactions in Operational Research*, 2(3):297–308.

Feillet, D., Dejax, P., and Gendreau, M. (2005). Traveling salesman problems with profits. *Transportation Science*, 39(2):188–205.

Ferreira, J., Quintas, A., Oliveira, J. A., Pereira, G. A. B., and Dias, L. (2014). *Solving the Team Orienteering Problem: Developing a Solution Tool Using a Genetic Algorithm Approach*, pages 365–375. Springer International Publishing, Cham.

Fischetti, M., Salazar-González, J. J., and Toth, P. (1995). The symmetric generalized traveling salesman polytope. *Networks*, 26(2):113–123.

Fischetti, M., Salazar-González, J. J., and Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45:378–394.

Fischetti, M., Salazar-González, J. J., and Toth, P. (1998). Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10:133–148.

García, S., Fernández, A., Luengo, J., and Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044 – 2064. Special Issue on Intelligent Distributed Information Systems.

Geem, Z. W., Tseng, C.-L., and Park, Y. (2005). Harmony search for generalized orienteering problem: Best touring in china. In Wang, L., Chen, K., and Ong, Y. S., editors, *Advances in Natural Computation*, pages 741–750, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gendreau, M., Hertz, A., and Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094.

Gendreau, M., Laporte, G., and Semet, F. (1998a). A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research*, 106(2):539–545.

Gendreau, M., Laporte, G., and Semet, F. (1998b). A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks*, 32:263–273.

Goldberg, A. V. and Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940.

Goldberg, A. V. and Tsioutsiouliklis, K. (2001). Cut tree algorithms: An experimental study. *Journal of Algorithms*, 38(1):51 – 83.

Golden, B. L., Levy, L., and Vohra, R. (1987). The orienteering problem. *Naval Research Logistics*, 34:307–318.

Golden, B. L., Wang, Q., and Liu, L. (1988). A multifaceted heuristic for the orienteering problem. *Naval Research Logistics (NRL)*, 35(3):359–366.

Gomory, R. and Hu, T. (1961). Multiterminal network flows. *Journal of The Society for Industrial and Applied Mathematics*, 9.

Grötschel, M. and Holland, O. (1991). Solution of large-scale symmetric travelling salesman problems. *Mathematical Programming*, 51(1):141–202.

Grötschel, M. and Padberg, M. (1979). On the symmetric travelling salesman problem I: Inequalities. *Mathematical Programming*, 16(1):265–280.

Gunawan, A., Lau, H., Vansteenwegen, P., and Lu, K. (2017). Well-tuned algorithms for the team orienteering problem with time windows. *Journal of the Operational Research Society*, 68.

Gunawan, A., Lau, H. C., and Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315 – 332.

Gusfield, D. (1990). Very simple methods for all pairs network flow analysis. *SIAM Journal on Computing*, 19(1):143–155.

Gutin, G. and Punnen, A. P. (2007). *The Traveling Salesman Problem and Its Variations (Combinatorial Optimization)*. Springer.

Hao, J. and Orlin, J. B. (1992). A faster algorithm for finding the minimum cut in a graph. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '92, pages 165–174. Society for Industrial and Applied Mathematics.

Hong, S. (1972). *A Linear Programming Approach for the Traveling Salesman Problem.* Ph.D. Thesis. Johns Hopkins University, Baltimore, Maryland, USA.

Ilhan, T., Iravani, S., and Daskin, M. (2008). The orienteering problem with stochastic profits. *IIE Transactions*, 40:406–421.

Jepsen, M. K., Petersen, B., Spoorendonk, S., and Pisinger, D. (2014). A branch-and-cut algorithm for the capacitated profitable tour problem. *Discrete Optimization*, 14:78 – 96.

Jünger, M., Rinaldi, G., and Thienel, S. (2000). Practical performance of efficient minimum cut algorithms. *Algorithmica*, 26:172–195.

Karp, R. M. (1972). *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA.

Keller, C. (1989). Algorithms to solve the orienteering problem: A comparison. *European Journal of Operational Research*, 41(2):224 – 231.

Keshtkaran, M., Ziarati, K., Bettinelli, A., and Vigo, D. (2015). Enhanced exact solution methods for the team orienteering problem. *International Journal of Production Research*, ahead-of-print:1–11.

Kobeaga, G., Merino, M., and Lozano, J. A. (2018). An efficient evolutionary algorithm for the orienteering problem. *Computers & Operations Research*, 90:42 – 59.

Kobeaga, G., Merino, M., and Lozano, J. A. (2020a). On solving cycle problems with branch-and-cut: Extending shrinking and exact subcycle elimination separation algorithms. arXiv:2004.14574.

Kobeaga, G., Merino, M., and Lozano, J. A. (2020b). A revisited branch-and-cut algorithm for large-scale orienteering problems. arXiv:2011.02743.

Labadie, N., Melechovský, J., and Calvo, R. (2011). Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. *J. Heuristics*, 17:729–753.

Laporte, G. and Martello, S. (1990). The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2):193 – 207.

Leifer, A. C. and Rosenwein, M. B. (1994). Strong linear programming relaxations for the orienteering problem. *European Journal of Operational Research*, 73(3):517–523.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44(10):2245–2269.

Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516.

Marinakis, Y., Politis, M., Marinaki, M., and Matsatsinis, N. (2015). *A Memetic-GRASP Algorithm for the Solution of the Orienteering Problem*, pages 105–116. Springer International Publishing, Cham.

Ostrowski, K., Karbowska-Chilinska, J., Koszelew, J., and Zabielski, P. (2017). Evolution-inspired local improvement algorithm solving orienteering problem. *Annals of Operations Research*, 253(1):519–543.

Padberg, M. and Grötschel, M. (1985). Polyhedral computations. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shimoys, editors, *The Traveling Salesman Problem*, pages 307–360. John Wiley & Sons, Chichester, UK.

Padberg, M. and Hong, S. (1980). *On the symmetric travelling salesman problem: A computational study*, pages 78–107. Springer Berlin Heidelberg, Berlin, Heidelberg.

Padberg, M. and Rinaldi, G. (1990a). An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming*, 47(1):19–36.

Padberg, M. and Rinaldi, G. (1990b). Facet identification for the symmetric traveling salesman polytope. *Mathematical Programming*, 47(1):219–257.

Padberg, M. and Sung, T.-Y. (1991). An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming*, 52(1):315–357.

Pferschy, U. and Staněk, R. (2017). Generating subtour elimination constraints for the tsp from pure integer solutions. *Central European Journal of Operations Research*, 25:231–260.

Poggi, M., Viana, H., and Uchoa, E. (2010). The Team Orienteering Problem: Formulations and Branch-Cut and Price. In Erlebach, T. and Lübbecke, M., editors, *10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'10)*, volume 14 of *OpenAccess Series in Informatics (OASIcs)*, pages 142–155, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Ramesh, R. and Brown, K. M. (1991). An efficient four-phase heuristic for the generalized orienteering problem. *Computers & Operations Research*, 18(2):151 – 165.

Ramesh, R., Yoon, Y.-S., and Karwan, M. H. (1992). An optimal algorithm for the orienteering tour problem. *ORSA Journal on Computing*, 4(2):155–165.

Reinelt, G. (1991). TSPLIB - a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384.

Riera-Ledesma, J. and Salazar-González, J. J. (2017). Solving the team orienteering arc routing problem with a column generation approach. *European Journal of Operational Research*, 262(1):14 – 27.

Santini, A. (2019). An adaptive large neighbourhood search algorithm for the orienteering problem. *Expert Systems with Applications*, 123:154 – 167.

Schilde, M., Doerner, K. F., Hartl, R. F., and Kiechle, G. (2009). Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, 3(3):179–201.

Silberholz, J. and Golden, B. (2010). The effective application of a new approach to the generalized orienteering problem. *Journal of Heuristics*, 16(3):393–415.

Souffriau, W., Vansteenwegen, P., Vertommen, J., Berghe, G. V., and Oudheusden, D. V. (2008). A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence archive*, 22:964–985.

Tasgetiren, M. F. (2001). A genetic algorithm with an adaptive penalty function for the orienteering problem. *Journal of Economic and Social Research*, 4(2):1–26.

Tsiligirides, T. (1984). Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35:797–809.

Vansteenwegen, P. and Gunawan, A. (2019). *Orienteering Problems: Models and Algorithms for Vehicle Routing Problems with Profits*. Springer International Publishing, Cham.

Vansteenwegen, P., Souffriau, W., and Oudheusden, D. V. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10.

Vansteenwegen, P., Souffriau, W., V. Berghe, G., and Van Oudheusden, D. (2009). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–3290.

Vansteenwegen, P. and Van Oudheusden, D. (2007). The mobile tourist guide: An or opportunity. *OR Insight*, 20(3):21–27.

Verbeeck, C., Sörensen, K., Aghezzaf, E.-H., and Vansteenwegen, P. (2014). A fast solution method for the time-dependent orienteering problem. *European Journal of Operational Research*, 236(2):419 – 432.

Wang, Q., Sun, X., Golden, B. L., and Jia, J. (1995). Using artificial neural networks to solve the orienteering problem. *Annals of Operations Research*, 61(1):111–120.

Wang, X., Golden, B. L., and Wasil, E. A. (2008). *Using a Genetic Algorithm to Solve the Generalized Orienteering Problem*, pages 263–274. Springer US, Boston, MA.

Whitley, D. L., Starkweather, T., and Fuquay, D. (1989). Scheduling problems and travelling salesman: The genetic edge recombination operator. In Schaffer, J. D., editor, *Proc. of the Third Int. Conf. on Genetic Algorithms*, pages 133–140, San Mateo, CA. Morgan Kaufmann.

Yuen, M., King, I., and Leung, K. (2011). A survey of crowdsourcing systems. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 766–773.

APPENDIX **A**

Pseudocodes

A.1 Shrinking and SEC Separation Strategies

In this appendix, we detail the pseudocodes of the shrinking and separation strategies used in the computational experiments for Chapter 3. These strategies are combinations of the shrinking rules proposed in Section 3.3.1 and Section 3.3.2, and the exact separation algorithms proposed in Section 3.4.

The pseudocodes should be considered as illustrations of the implementations of strategies whose aim is to help the reader to understand how the strategies work. The source code in C of the computational implementations is available at `https://github.com/gkobeaga/cpsrksec`. In Table A.1, we detail the meaning of the symbols used in the pseudocodes.

A.1.1 Shrinking Strategies

The shrinking strategies are combinations of the shrinking rules of Section 3.3.1 and Section 3.3.2. In total, 5 different shrinking strategies for SECs are obtained: C1, C1C2, C1C2C3, S1 and S1S2. The SHRINK/UPDATE procedure refers to a process performed every time a set is shrunk.

Symbol		Meaning
$G = (V, E)$		Input graph of the cycle problem
$G^* = (V^*, E^*)$		Support graph
(y, x)	$\in P_A^G$	A solution of the LP_0
m	$\in \mathbb{R}_+^{V^*}$	A vector where $m_v = \max\{y_u : u \in \pi(v)\}$
H	$\subset V^*$	Heap: vertices remaining to check
S	$\subset V^*$	A subset candidate for the shrinking
Q	$\subset V$	A subset of V
$\mathcal{Q}$	$\subset \mathcal{P}(V)$	List of Q sets of V
$\mathcal{L}$		List of violated SECs
D	$\subset V^*$	Set of fixed vertices. In a cycle problem with depot: $D = \{d\}$
O	$\subset V^*$	Set of vertices with value one
$(k_{in} \times k_{out})$	$\in \mathbb{N}_+ \times \mathbb{N}_+$	Maximum vertices (inside and outside) considered when generating the violated SECs from the Q sets
$T = (V, A_T)$		A directed rooted tree
$parent$	$V \rightarrow V$	Successive parent of each v in the tree
$child$	$V \rightarrow V$	Successive children of each v in the tree
w	$\in \mathbb{R}_+^{A_T}$	Weights of the arcs of the Gomory-Hu tree
$\bar{G} = (\bar{V}, \bar{E})$		Generic graph used in the Gomory-Hu tree construction

Table A.1: A summary of the symbols used in the pseudocodes

Algorithm SHRINK/UPDATE: Shrink graph and vectors. Save Q sets. Update heap.

input : G^*, (y, x), m, H, S and $\mathcal{Q}$
output: G^*, (y, x), m, H, s and $\mathcal{Q}$

1 $G^* \leftarrow G^*[S]$;
2 $(y, x) \leftarrow (y[S], x[S])$;
3 $m \leftarrow m[S]$;
4 $H \leftarrow H[S]$;
5 $O \leftarrow \{v \in V^* : m_v \geq 1\}$;
6 **for** $n \in N(s)$ **do**
7   **if** $y_n < x_{[n,s]}$ **then**
8   **for** $r \in O$ **do**
9   **if** $r \neq s$ **then**
10   **if** $\langle \{s, n\}, s, r \rangle$ *violates* (3.35) **then**
11                     $Q \leftarrow \{\pi(\{s, n\})\}$;
12   **if** $|Q| > |V|/2$ **then**
13                         $Q \leftarrow V - Q$;
14   **end**
15                     $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{Q\}$;
16   **goto** line 20;
17   **end**
18   **end**
19   **end**
20   **end**
21     $H \leftarrow H \cup \{n\}$;
22 **end**

Algorithm C1: Shrinking: Rule C1

```
   input  : G*, (y, x), m, H and Q
   output: G*, (y, x), m, H and Q
1  while |H| ≠ ∅ do
2      Select a vertex u ∈ H;
3      H ← H − {u};
4      c ← y_u;
5      for v ∈ N(u) do
6          if y_v = c and x_[u,v] = c then
7              for t ∈ N(v) − {u} do
8                  if y_t = c and x_[v,t] = c then
9                      S ← {u, v};
10                     SHRINK/UPDATE (G*, (y, x), m, H, S, Q);
11                     goto line 15;
12                 end
13             end
14         end
15     end
16 end
```

Algorithm C1C2: Shrinking: Rule C1 and Rule C2

```
   input  : G*, (y, x), m, H and Q
   output: G*, (y, x), m, H and Q
1  while |H| ≠ ∅ do
2      Select a vertex u ∈ H;
3      H ← H − {u};
4      c ← y_u;
5      for v ∈ N(u) do
6          if y_v = c and x_[u,v] = c then
7              for t ∈ N(v) − {u} do
8                  if y_t = c and x_[u,t] + x_[v,t] = c then
9                      S ← {u, v};
10                     SHRINK/UPDATE (G*, (y, x), m, H, S, Q);
11                     goto line 15;
12                 end
13             end
14         end
15     end
16 end
```

Algorithm C1C2C3: Shrinking: Rule C1, C2 and C3

input : G^*, (y, x), m, H and $\mathcal{Q}$
output: G^*, (y, x), m, H and $\mathcal{Q}$

1 **while** $|H| \neq \emptyset$ **do**
2 Select a vertex $u \in H$;
3 $H \leftarrow H - \{u\}$;
4 $c \leftarrow y_u$;
5 **for** $v \in N(u)$ **do**
6   **if** $y_v = c$ *and* $x_{[u,v]} = c$ **then**
7   **for** $t \in N(v) - \{u\}$ **do**
8   **if** $y_t = c$ *and* $x_{[u,t]} + x_{[v,t]} = c$ **then**
9      $S \leftarrow \{u, v\}$;
10   SHRINK/UPDATE $(G^*, (y, x), m, H, S, \mathcal{Q})$;
11   **goto** line 26;
12   **end**
13   **end**
14   **for** $w \in N(v) - \{u\}$ **do**
15   **if** $x_{[u,t]} + x_{[u,w]} + x_{[v,w]} = 2c$ **then**
16   **for** $t \in N(w) - \{v, u\}$ **do**
17   **if** $y_t = c$ *and* $x_{[u,t]} + x_{[v,t]} = c$ **then**
18        $S \leftarrow \{u, v, w\}$;
19   SHRINK/UPDATE $(G^*, (y, x), m, H, S, \mathcal{Q})$;
20   **goto** line 26;
21   **end**
22   **end**
23   **end**
24   **end**
25   **end**
26 **end**
27 **end**

Algorithm S1: Shrinking: Rule S1

input : G^*, (y, x), m, H and $\mathcal{Q}$
output: G^*, (y, x), m, H and $\mathcal{Q}$

1 **while** $|H| \neq \emptyset$ **do**
2   Select a vertex $u \in H$;
3     $H \leftarrow H - \{u\}$;
4     $c \leftarrow y_u$;
5   **for** $v \in N(u)$ **do**
6   **if** $y_v = c$ *and* $x_{[u,v]} = c$ **then**
7   **if** $\exists w \in V^* - \{u, v\}$ *such that* $y_w \geq c$ **then**
8                 $S \leftarrow \{u, v\}$;
9   SHRINK/UPDATE $(G^*, (y, x), m, H, S, \mathcal{Q})$;
10   **goto** line 13;
11   **end**
12   **end**
13   **end**
14 **end**

Algorithm S1S2: Shrinking: Rule S1 and S2

input : G^*, (y, x), m, H, D and $\mathcal{Q}$
output: G^*, (y, x), m, H, D and $\mathcal{Q}$

1 **while** $|H| \neq \emptyset$ **do**
2   Select a vertex $u \in H$;
3     $H \leftarrow H - \{u\}$;
4     $c \leftarrow y_u$;
5   **for** $v \in N(u)$ **do**
6   **if** $y_v = c$ *and* $x_{[u,v]} = c$ **then**
7   **if** $\exists w \in V^* - \{u, v\}$ *such that* $y_w \geq c$ **then**
8                 $S \leftarrow \{u, v\}$;
9   SHRINK/UPDATE $(G^*, (y, x), m, H, S, \mathcal{Q})$;
10   **goto** line 17;
11   **end**
12   **else if** $x_{[u,v]} > y_u$ *and* $x_{[u,v]} > y_v$ **then**
13             $S \leftarrow \{u, v\}$;
14   SHRINK/UPDATE $(G^*, (y, x), m, H, S, \mathcal{Q})$;
15   **goto** line 17;
16   **end**
17   **end**
18 **end**

A.1.2 Exact SEC Separation Strategies

The exact separation strategies detailed in this appendix refer to the separation algorithms used for the experiments in Section 3.5. We assume that the vertex set $V^* = \{v_1, \ldots, v_{|V^*|}\}$ is an ordered set. The CUTGEN algorithm is the procedure detailed in Section 3.5 to generate the most violated SECs corresponding to set Q given the parameter $(k_{in}, k_{out}) \in \mathbb{N}_+ \times \mathbb{N}_+$. The vector (k_{in}, k_{out}) represents the maximum amount of vertices that are considered inside and outside Q. Note that, CUTGEN is defined to select, for each inside vertex, a number of k_{out} different random outside vertices to maximize the randomness of the obtained violated SECs.

Algorithm EH: Extended Hong's exact separation algorithm

input : G^*, (y, x), D and (k_{in}, k_{out})
output: A list $\mathcal{L}$ of violated SECs

$V^* \leftarrow$ sort V^* decreasingly by y; $m \leftarrow y$;
$H \leftarrow V^*$;
Apply shrinking strategy $(G^*, (y, x), m, H, D, \mathcal{Q})$;
while $|V^*| > 1$ **do**
  $Q \leftarrow (v_1, v_2)$-minimum cut in the graph G^*;
  if $\langle Q, v_1, v_2 \rangle$ *violates* (3.35) **then**
    if $|Q| > |V|/2$ **then**
      $Q \leftarrow V - Q$;
    end
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\pi(Q)\}$;
  end
end
$\mathcal{L} \leftarrow$ CUTGEN $(G^*, (y, x), D, \mathcal{Q}, (k_{in}, k_{out}))$;

Algorithm DH: Dynamic Hong's exact separation algorithm

```
input  : G*, (y, x), D and (k_in, k_out)
output: A list ℒ of violated SECs
V* ← sort V* decreasingly by y;
m ← y;
H ← V*;
Apply shrinking strategy (G*, (y, x), m, H, 𝒬);
while |V*| > 1 do
    Q ← (v1, v2)-minimum cut in the graph G*;
    if ⟨Q, v1, v2⟩ violates (3.35) then
        if |Q| > |V|/2 then
            Q ← V − Q;
        end
        𝒬 ← 𝒬 ∪ {π(Q)};
    end
    if x[v1,v2] > y_v2 then
        reorder ← 1;
    else
        reorder ← 0;
    end
    S ← {v1, v2};
    G* ← G*[S];
    (y, x) ← (y[S], x[S]);
    m ← m[S];
    if reorder then
        V* ← sort V* decreasingly by y;
    end
end
ℒ ← CUTGEN (G*, (y, x), D, 𝒬, (k_in, k_out));
```

Algorithm DHI: Dynamic Hong with extra shrinking separation algorithm

input : G^*, (y, x), D and (k_{in}, k_{out})
output: A family $\mathcal{Q}$ of violated SECs

```
V* ← sort V* decreasingly by y;
m ← y;
H ← V*;
Apply shrinking strategy (G*, (y, x), m, H, 𝒬);
while |V*| > 1 do
    Q ← (v1, v2)-minimum cut in the graph G*;
    if ⟨Q, v1, v2⟩ violates (3.35) then
        if |Q| > |V|/2 then
            Q ← V − Q;
        end
        𝒬 ← 𝒬 ∪ {π(Q)};
    end
    if x_[v1,v2] > y_v2 then
        reorder ← 1;
    else
        reorder ← 0;
    end
    S ← {v1, v2};
    SHRINK/UPDATE (G*, (y, x), m, H, S, 𝒬);
    Apply shrinking strategy (G*, (y, x), m, H, 𝒬);
    if reorder then
        V* ← sort V* decreasingly by y;
    end
end
ℒ ← CUTGEN (G*, (y, x), D, 𝒬, (k_in, k_out));
```

Algorithm EPG: Extended Padberg-Grötschel exact separation algorithm

input : G^*, (y, x), D and (k_{in}, k_{out})
output: A family $\mathcal{Q}$ of violated SECs

$V^* \leftarrow$ sort V^* decreasingly by y;
$m \leftarrow y$;
Apply shrinking strategy $(G^*, (y, x), m, H, \mathcal{Q})$;
$(T, w, u) \leftarrow$GHTREE $(G^*, (y, x), v_1)$;
for $a \in A_T$ **do**
    $Q \leftarrow d_a$;
    if $w_a - 2 \cdot u_a - 2 \cdot v_a < 2$ **then**
        if $|Q| > |V|/2$ **then**
            $Q \leftarrow V - Q$;
        end
        $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\pi(Q)\}$;
    end
end
$\mathcal{L} \leftarrow$ CUTGEN $(G^*, (y, x), D, \mathcal{Q}, (k_{in}, k_{out}))$;

Algorithm CUTGEN: SEC generation

input : G^*, (y, x), D, $\mathcal{Q}$, (k_{in}, k_{out})
output: A family $\mathcal{L}$ of violated SECs

for $Q \in \mathcal{Q}$ **do**
  if $D \cap Q = \emptyset$ **then**
    $M_{in} \leftarrow \{v \in Q : y_v \geq y_u \ \forall u \in Q\}$;
    $S_{in} \leftarrow$ randomly select k_{in} vertices from M_{in};
  else
    $S_{in} \leftarrow$ a vertex in $D \cap Q$;
  end
  if $D - Q = \emptyset$ **then**
    $M_{out} \leftarrow \{v \in V^* - Q : y_v \geq y_u \ \forall u \in V^* - Q\}$;
  else
    $S_{out} \leftarrow$ a vertex in $D - Q$;
  end
  for $u \in S_{in}$ **do**
    if $D - Q = \emptyset$ **then**
      $S_{out} \leftarrow$ randomly select k_{out} vertices from M_{out};
    end
    for $v \in S_{out}$ **do**
      Add the violated SEC $\langle Q, u, v \rangle$ to $\mathcal{L}$;
    end
  end
end
$\mathcal{L} \leftarrow$ CUTGEN $(G^*, (y, x), D, \mathcal{Q}, (k_{in}, k_{out}))$;

A.1.3 Directed Rooted Gomory-Hu Tree

As was explained in Section 3.4, the key for an efficient extension of the Padberg-Grötschel exact separation algorithm is the construction of the directed rooted Gomory-Hu tree, which is detailed in the following pseudocodes. The novelty is the ADD-ARC/REORDER-TREE procedure, where we show how the Gomory-Hu construction must be adapted to evaluate the u_v values ($u_v = \arg\max\{y_u : u \in \Delta(v)\}$) and reorder the tree in order to maintain a given vertex in the top of the tree.

Algorithm GHTREE: Rooted directed Gomory-Hu tree

---

input : $\bar{G}$, (y, x), r
output: T, w, u: a rooted directed weighted tree

$T \leftarrow (V, \emptyset)$;
for $v \in V$ **do**
  $u_v = m_v = \arg\max\{y_w : w \in \pi(v) \in G^*\}$;
end
$\bar{G} \leftarrow G^*$ and consider $|\pi(v)| = 1$ for every $v \in \bar{V}$;
$(T, w, u) \leftarrow$ GHTREE-RECURSIVE$(\bar{G}, (y, x), r, T, w, u)$;

Algorithm GHTREE-RECURSIVE: Recursive operator to build the Gomory-Hu tree

---

input : $\bar{G}$, (y, x), r, T, w, u
output: T, w, u

$C \leftarrow \{v \in \bar{V} : |\pi(v)| = 1\}$;
if $|C| > 1$ **then**
  $(a, b) \leftarrow$ randomly select two different vertices from C;
  $(A : B) \leftarrow (a, b)$-minimum cut in $\bar{G}$;
  $(T, w, u, r_a, r_b) \leftarrow$ ADD-ARC/REORDER-TREE$(T, (y, x), m, u, r, A, B)$;
  $(T, w, u) \leftarrow$ GHTREE-RECURSIVE$(G^*[B], (y[B], x[B]), r_a, T, w, u)$;
  $(T, w, u) \leftarrow$ GHTREE-RECURSIVE$(G^*[A], (y[A], x[A]), r_b, T, w, u)$;
end

Algorithm ADD-ARC/REORDER-TREE: Add arc and reorder the tree

input : T, (y, x), m, u, r, A, B
output: T, w, u, r_a, r_b

```
if r ∈ A then
    r_a ← r;
    r_b ← b;
    if parent(r) ∈ A or parent(r) = ∅ then
        e = (r, b);
    else
        e = (b, r);
        f = (p(r), r);
        g = (p(r), b);
        w_g ← w_f;
        A_T = A_T − {f} ∪ {g};
        m_r = max{m_r, m_b};
    end
    u_r = m_r;
    u_b = m_b;
    for c ∈ child(r) do
        if c ∈ A then
            u_r = max{u_r, u_c};
        else
            A_T = A_T − {(r, c)} ∪ {(a, c)};
            u_b = max{u_b, u_c};
        end
    end
else
    r_a ← a;
    r_b ← r;
    if parent(r) ∈ B or parent(r) = ∅ then
        e = (r, a);
    else
        e = (a, r);
        f = (p(r), r);
        g = (p(r), a);
        w_g ← w_f;
        A_T = A_T − {f} ∪ {g};
    end
    u_r = m_r;
    u_a = m_a;
    for c ∈ child(r) do
        if c ∈ B then
            u_r = max{u_r, u_c};
        else
            A_T = A_T − {(r, c)} ∪ {(a, c)};
            u_a = max{u_a, u_c};
        end
    end
end
A_T = A_T ∪ {e};
w_e ← x(A : B);
```

APPENDIX **B**

Detailed Computational Results

B.1 Chapter 2: Evolutionary Algorithm

B.1.1 Initialization Parameter

We detail the results of Section 2.4.1, where the influence of the parameter p on the population initialization and on EA4OP is checked. Three different choices of p are tested: α^2, α and $\sqrt{\alpha}$ where $\alpha = d_0/v(TSP)$.

Table B.1: Initialization and EA4OP results in generation 1, depending on $p \in \{\alpha^2, \alpha, \sqrt{\alpha}\}$, where $\alpha = d_0/v(TSP)$.

instname	Gap						Time						Number of visited nodes						
	Initialization			EA4OP			Initialization			EA4OP			Initialization			EA4OP			B&C
	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	
gil262	28.29	22.22	18.67	2.91	2.22	3.23	0.45	0.41	0.55	3.00	3.27	3.14	113.30	122.90	128.50	153.40	154.50	152.90	158
a280	22.59	17.82	13.06	5.10	4.49	6.94	0.57	0.50	0.65	2.59	2.63	2.53	113.80	120.80	127.80	139.50	140.40	136.80	147
pr299	26.48	17.78	13.64	3.52	2.96	2.84	0.66	0.62	0.80	2.90	3.07	2.69	119.10	133.20	139.90	156.30	157.20	157.40	162
lin318	32.49	25.96	24.49	4.20	3.79	4.83	0.79	0.78	1.02	7.83	7.62	7.07	138.40	151.80	154.80	196.40	197.20	195.10	205
rd400	26.03	22.89	20.00	3.85	2.68	3.64	1.01	0.87	1.32	7.93	8.14	6.37	176.80	184.30	191.20	229.80	232.60	230.30	239
pcb3038	22.88	20.54	14.68	2.02	2.96	1.66	53.81	122.50	298.12	468.62	515.92	645.39	1220.80	1257.80	1350.60	1551.10	1536.10	1556.80	.
fl3795	32.99	27.82	22.71	2.50	2.29	1.09	199.38	462.18	831.46	2665.55	2939.52	4538.55	1217.60	1311.50	1404.40	1771.60	1775.30	1797.20	.
fnl4461	16.12	21.85	15.34	1.63	0.36	0.60	201.57	328.56	909.11	1914.74	2094.99	2805.33	1951.10	1817.70	1969.20	2288.10	2317.60	2312.00	.
rl5934	32.82	21.62	14.10	4.08	4.41	1.53	552.88	1410.71	2648.46	3925.22	4451.34	6476.74	2127.50	2482.20	2720.40	3037.70	3027.40	3118.60	.
pla7397	48.73	32.53	20.41	7.06	6.61	3.58	341.21	1347.14	3256.91	15276.42	-	-	2665.90	3508.30	4138.60	4832.70	4856.50	5013.90	

Table B.2: Initialization and EA4OP results in generation 2, depending on $p \in \{\alpha^2, \alpha, \sqrt{\alpha}\}$, where $\alpha = d_0/v(TSP)$.

instname	Gap						Time						Number of visited nodes						
	Initialization			EA4OP			Initialization			EA4OP			Initialization			EA4OP			B&C
	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	
gil262	32.76	22.28	16.89	3.34	2.30	3.25	0.43	0.44	0.59	3.80	3.32	3.39	100.50	112.10	116.70	134.50	134.70	132.20	133
a280	27.06	18.15	14.16	2.32	1.57	3.44	0.57	0.56	0.71	3.00	2.85	2.61	109.40	116.80	121.60	132.30	132.70	131.30	135
pr299	25.69	17.58	12.53	1.55	2.13	2.40	0.66	0.67	0.85	3.99	3.51	3.57	120.00	127.70	133.30	147.80	146.20	145.10	148
lin318	33.15	23.18	18.72	3.07	2.22	2.06	0.81	0.81	1.06	8.41	7.39	7.89	125.90	142.20	150.90	180.30	183.80	184.40	189
rd400	29.98	25.75	19.80	2.66	2.61	2.73	0.94	0.92	1.38	7.81	7.78	7.79	171.90	169.20	179.90	213.40	214.00	214.20	218
pcb3038	30.38	24.00	17.16	1.09	1.68	0.95	50.66	141.91	323.32	482.04	568.30	738.34	1181.10	1187.40	1261.60	1468.40	1453.50	1471.00	.
fl3795	40.86	29.24	25.73	6.87	3.69	2.55	211.15	482.52	853.80	3074.42	5410.46	5654.73	1061.50	1213.90	1288.00	1630.20	1682.20	1696.80	.
fnl4461	26.18	27.70	20.03	2.50	1.94	1.53	185.30	387.57	984.93	1971.86	2481.72	2827.26	1871.60	1695.20	1824.10	2155.90	2152.90	2153.50	.
rl5934	34.97	21.88	16.61	4.25	4.93	3.23	584.43	1488.55	2752.74	3833.70	4823.62	5862.78	2030.90	2384.80	2550.40	2885.10	2872.90	2959.80	.
pla7397	49.14	31.72	21.36	3.68	2.05	2.38	370.41	1637.00	3973.10	-	-	-	2593.40	3285.80	3674.00	4375.60	4444.60	4434.80	

Table B.3: Initialization and EA4OP results in generation 3, depending on $p \in \{\alpha^2, \alpha, \sqrt{\alpha}\}$, where $\alpha = d_0/v(TSP)$.

	Gap						Time						Number of visited nodes						
	Initialization			EA4OP			Initialization			EA4OP			Initialization			EA4OP			B&C
instname	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	
gil262	31.01	20.96	15.31	2.79	2.05	2.07	0.44	0.43	0.57	4.06	3.66	3.80	104.50	115.90	124.10	143.40	143.80	143.60	148
a280	30.35	26.04	20.78	12.91	12.55	12.30	0.56	0.52	0.66	3.44	3.44	3.40	115.50	118.80	121.60	133.80	134.30	132.20	139
pr299	29.98	22.19	15.20	4.41	4.43	3.96	0.65	0.62	0.82	4.71	4.67	4.47	115.70	124.80	130.20	143.40	142.50	144.80	149
lin318	34.76	27.90	22.21	3.78	3.03	2.04	0.76	0.78	1.05	7.70	7.42	6.46	127.10	136.50	143.00	178.90	180.10	182.90	193
rd400	29.61	25.38	17.82	2.18	1.91	1.55	0.98	0.92	1.39	8.26	7.43	7.36	175.90	176.40	184.20	218.90	217.00	218.20	223
pcb3038	37.37	28.49	19.07	3.05	1.86	1.14	51.42	129.64	302.42	828.24	903.64	1126.98	1194.90	1240.90	1342.90	1568.00	1578.60	1576.60	.
fl3795	33.67	30.87	23.46	3.87	3.20	2.26	229.70	488.03	861.10	2231.22	3077.29	3711.79	1146.10	1217.20	1305.70	1666.80	1668.60	1654.50	.
fnl4461	30.52	29.89	18.97	1.98	1.69	0.92	190.53	359.64	933.53	3054.00	2903.93	3163.09	1897.80	1742.50	1927.90	2257.10	2251.60	2246.30	.
rl5934	39.99	24.10	17.01	8.29	4.51	2.79	578.97	1474.82	2707.07	4126.14	5053.15	6080.23	2106.30	2511.40	2671.90	2980.90	3062.20	3078.90	.
pla7397	53.86	33.00	21.55	1.02	1.10	0.88	348.59	1497.58	3733.19	17495.11	16255.44	17604.69	2563.80	3436.60	3825.40	4790.20	4774.50	4742.10	.

Table B.4: Initialization and EA4OP results in generation 4, depending on $p \in \{\alpha^2, \alpha, \sqrt{\alpha}\}$, where $\alpha = d_0/v(TSP)$.

	Gap						Time						Number of visited nodes						
	Initialization			EA4OP			Initialization			EA4OP			Initialization			EA4OP			B&C
instname	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	$p=\alpha^2$	$p=\alpha$	$p=\sqrt{\alpha}$	
gil262	17.71	12.51	16.69	2.21	0.96	3.39	0.67	0.45	0.59	1.65	1.48	1.48	31.10	35.30	30.50	37.00	37.00	34.40	36
a280	17.09	9.69	5.58	0.52	0.57	0.31	0.39	0.47	0.58	4.08	3.51	4.19	181.30	189.50	197.30	202.60	203.30	204.20	204
pr299	3.96	1.31	0.84	0.05	0.04	0.03	0.40	0.46	0.48	5.71	4.67	5.17	273.50	276.70	278.20	280.40	281.20	280.00	280
lin318	11.51	7.09	3.88	0.23	0.15	0.54	0.57	0.67	0.79	9.78	10.70	7.86	249.60	257.40	265.60	277.90	277.30	276.10	280
rd400	4.06	2.32	0.96	0.08	0.03	0.05	0.46	0.59	0.70	9.66	12.03	12.14	375.60	372.40	373.70	382.90	382.70	382.50	382
pcb3038	27.52	23.07	16.28	1.48	1.40	1.10	61.75	141.89	314.50	631.88	694.24	840.03	1344.80	1335.90	1420.10	1628.00	1630.00	1634.50	.
fl3795	40.88	36.28	28.16	4.54	1.75	2.62	157.26	251.02	573.68	3293.72	3752.92	4276.60	811.90	822.40	921.50	1233.30	1245.80	1249.70	.
fnl4461	24.08	37.84	28.53	2.35	2.50	2.67	251.78	224.61	734.87	1139.14	981.03	1530.37	1099.90	873.00	957.30	1275.10	1278.90	1260.00	.
rl5934	48.66	32.45	20.80	8.93	9.09	2.46	423.93	981.89	2181.46	2014.67	2822.77	4658.60	1313.90	1664.40	1932.30	2206.80	2186.30	2384.50	.
pla7397	58.45	35.30	26.62	1.82	6.00	6.24	405.62	859.67	2853.34	3734.00	4771.74	7490.28	1106.90	1606.50	1804.20	2240.80	2194.20	2214.20	.

B.1.2 Contribution of the genetic components

We detail the results of Section 2.4.1, where the contribution of the components in the EA4OP algorithm are evaluated.

Table B.5: Results of Algorithm 3.3.1, Algorithm 3.3.2, Algorithm 3.3.3 and EA4OP in generation 1.

	Algorithm 3.3.1			Algorithm 3.3.2			Algorithm 3.3.3			EA4OP		
instance	best	gap	time	best	gap	time	best	gap	time	best	gap	time
gil262	134	15.19	5.82	136	13.92	1.09	139	12.03	3.03	156	1.27	2.83
a280	133	9.52	7.10	134	8.84	1.02	136	7.48	3.02	143	2.72	3.00
lin318	171	16.59	12.40	184	10.24	3.38	185	9.76	7.07	202	1.46	7.15
pr299	144	11.11	8.01	145	10.49	1.39	147	9.26	3.02	160	1.23	3.12
rd400	200	16.32	18.69	215	10.04	2.17	216	9.62	7.04	234	2.09	6.59
pcb3038	1365	13.17	8903.48	1401	10.88	304.07	1437	8.59	681.24	1572	*	681.94
fl3795	1496	17.58	14438.13	1616	10.96	670.28	1669	8.04	2996.22	1815	*	2994.90
fnl4461	1993	15.19	–	2097	10.77	1024.42	2172	7.57	2463.64	2350	*	2462.65
rl5934	2784	11.48	–	2982	5.18	2445.80	3051	2.99	5383.43	3145	*	5382.25
pla7397	4188	18.54	–	4495	12.57	3195.25	4628	9.98	15982.47	5141	*	15981.78

Table B.6: Results of Algorithm 3.3.1, Algorithm 3.3.2, Algorithm 3.3.3 and EA4OP in generation 2.

	Algorithm 3.3.1			Algorithm 3.3.2			Algorithm 3.3.3			EA4OP		
instance	best	gap	time	best	gap	time	best	gap	time	best	gap	time
gil262	7201	13.46	8.16	7611	8.53	1.23	7630	8.30	4.03	8175	1.75	3.47
a280	7411	12.07	8.68	7494	11.08	1.09	7515	10.83	3.03	8304	1.47	2.85
lin318	9297	14.89	14.08	10362	5.14	2.74	10439	4.43	8.07	10866	0.52	8.29
pr299	8418	8.32	9.20	8652	5.77	1.46	8698	5.27	3.03	9112	0.76	3.23
rd400	11295	17.26	19.70	11670	14.52	2.23	11836	13.30	7.05	13442	1.54	6.80
pcb3038	77315	15.82	12439.37	80334	12.53	331.58	83847	8.71	820.23	91842	*	820.37
fl3795	87534	15.34	–	93116	9.94	748.88	97617	5.59	4789.09	103397	*	4788.96
fnl4461	113951	18.85	–	122232	12.96	1014.76	128427	8.54	2619.03	140424	*	2618.15
rl5934	146403	14.71	–	157466	8.26	2591.68	166807	2.82	5757.77	171649	*	5757.80
pla7397	226347	16.92	–	244388	10.30	3919.61	261568	3.99	–	272452	*	–

Table B.8: Results of Algorithm 3.3.1, Algorithm 3.3.2, Algorithm 3.3.3 and EA4OP in generation 4.

	Algorithm 3.3.1			Algorithm 3.3.2			Algorithm 3.3.3			EA4OP		
instance	best	gap	time	best	gap	time	best	gap	time	best	gap	time
gil262	1955	3.74	3.43	2004	1.33	1.07	2004	1.33	2.02	2030	0.05	1.35
a280	11615	3.72	7.11	11681	3.17	1.41	11714	2.90	4.04	12048	0.13	3.39
lin318	14739	2.60	10.39	14911	1.46	2.59	14892	1.59	8.07	15119	0.09	7.91
pr299	14954	0.21	3.63	14947	0.26	1.70	14956	0.20	4.06	14980	0.04	3.46
rd400	19994	0.56	9.88	20071	0.18	2.23	20071	0.18	10.10	20101	0.03	9.61
pcb3038	87338	13.67	13477.94	89617	11.42	331.67	92835	8.24	800.34	101173	*	800.13
fl3795	69006	13.82	–	72665	9.25	671.90	75807	5.32	4496.88	80069	*	4496.09
fnl4461	64382	24.33	–	71304	16.20	796.02	74942	11.92	1490.72	85088	*	1490.80
rl5934	118749	13.85	–	125856	8.69	2603.21	130007	5.68	4038.32	137838	*	4037.07
pla7397	116662	18.07	–	130276	8.51	3051.93	135336	4.96	6667.88	142399	*	6667.36

Table B.7: Results of Algorithm 3.3.1, Algorithm 3.3.2, Algorithm 3.3.3 and EA4OP in generation 3.

	Algorithm 3.3.1			Algorithm 3.3.2			Algorithm 3.3.3			EA4OP		
instance	best	gap	time	best	gap	time	best	gap	time	best	gap	time
gil262	8274	10.51	8.21	8429	8.84	1.27	8708	5.82	4.04	9094	1.64	3.94
a280	8001	18.14	8.98	8117	16.95	1.06	8229	15.81	4.02	8684	11.15	3.22
lin318	8484	18.17	12.13	9625	7.17	3.16	9625	7.17	7.09	10273	0.92	6.33
pr299	9071	12.30	10.35	9146	11.57	1.47	9239	10.67	4.04	9959	3.71	3.95
rd400	11400	13.79	22.63	11625	12.09	2.78	11779	10.92	8.05	13088	1.02	7.74
pcb3038	88097	15.83	16178.25	88756	15.20	309.60	92394	11.73	917.28	104667	*	917.39
fl3795	82427	15.64	–	91545	6.31	824.38	92140	5.70	3160.52	97707	*	3158.89
fnl4461	135326	17.59	–	142804	13.03	956.08	149330	9.06	3248.98	164201	*	3248.64
rl5934	172220	16.96	–	193989	6.46	2831.70	193768	6.57	5882.33	207385	*	5881.87
pla7397	257454	19.73	–	276725	13.72	3673.65	299270	6.70	–	320744	*	–

Table B.9: Contribution of the k-d tree based add operator: Generation 1

	Cheapest insertion		3-nearest insertion (using k-d trees)	
instname	Best	Time	Best	Time
gil262	**157**	4.29	156	**2.84**
a280	140	3.27	**143**	**3.00**
pr299	160	4.32	160	**3.12**
lin318	202	7.42	202	**7.15**
rd400	**236**	12.35	234	**6.59**
pcb3038	**1608**	3014.56	1572	**681.94**
fl3795	1798	8105.06	**1815**	**2994.90**
fnl4461	2326	8883.04	**2350**	**2462.65**

B.1.3 Add operator

In this section we detail the preliminary experiments carried out for the add operator. In tables B.9, B.10 and B.11 we show the contribution of the 3-nearest insertion approach, which uses the k-d trees, in relation to the cheapest insertion heuristic. The headings are as follows: *instance*, name codification of the instance; *best*, best known solution of the corresponding algorithm; *time*, average time (in seconds) of 10 runs. In the last row, average summary for gap and time are shown.

B.1.4 Comparison with state-of-the-art Algorithms

In this Appendix the numerical results are detailed for the four algorithms (B&C, 2-P IA, GRASP-PR and EA4OP) and the full classification, that is, eight tables. Table B.12

Table B.10: Contribution of the k-d tree based add operator: Generation 2

	Cheapest insertion		3-nearest insertion (using k-d trees)	
instname	Best	Time	Best	Time
gil262	**8266**	4.47	8175	**3.47**
a280	8301	3.93	**8304**	**2.85**
pr299	**9115**	5.27	9112	**3.23**
lin318	**10901**	9.74	10866	**8.29**
rd400	**13576**	12.94	13442	**6.80**
pcb3038	**92353**	3208.56	91842	**820.37**
fl3795	**104503**	11156.20	103397	**4788.96**
fnl4461	140361	10222.61	**140424**	**2618.15**

Table B.11: Contribution of the k-d tree based add operator: Generation 3

	Cheapest insertion		3-nearest insertion (using k-d trees)	
instname	Best	Time	Best	Time
gil262	**9124**	4.93	9094	**3.94**
a280	**8695**	4.86	8684	**3.22**
pr299	**10120**	5.81	9959	**3.95**
lin318	**10339**	7.70	10273	**6.33**
rd400	**13122**	12.31	13088	**7.74**
pcb3038	**106347**	3494.96	104667	**917.39**
fl3795	**98394**	10604.92	97707	**3158.89**
fnl4461	163465	10030.42	**164201**	**3248.64**

shows the results for generation 1 and medium-sized instances, Table B.13 for generation 1 and large-sized instances, Table B.14 for generation 2 and medium-sized instances, Table B.15 for generation 2 and large-sized instances, Table B.16 for generation 3 and medium-sized instances, Table B.17 for generation 3 and large-sized instances, Table B.18 for generation 4 and medium-sized instances and Table B.19 for generation 4 and large-sized instances. The headings are as follows: *instance*, name codification of the instance; *best*, best known solution of the corresponding algorithm; *gap*, quality gap with respect to the global best known solution; *time*, average time (in seconds) of 10 runs. In the last row, average summary for gap and time are shown. The symbols mean the following: $*$, best known solution achieved (or optimum solution achieved for instances in which B&C finishes before time limit); $-$, execution stopped because 5-hour time limit was exceeded; NA, solution not available after time limit exceeded; “ . ”, the code finished unexpectedly. The best results for each instance among heuristics are in **bold**, in terms of quality solution and time. In the last row of the tables, average gap and average time are computed, considering 18000 seconds for problems that did not finish in that time. The averages are calculated excluding missing values.

Table B.12: Generation 1, $n \leq 400$

				Branch-&-Cut			2-Parameter IA			GRASP with PR			EA4OP		
instance	d0	α	opt	best	gap	time	best	gap	time	best	gap	time	best	gap	time
att48	5314	0.50	31	31	*	0.00	**31**	*	**0.07**	**31**	*	0.15	**31**	*	0.25
gr48	2523	0.50	31	31	*	0.00	**31**	*	**0.07**	**31**	*	0.13	**31**	*	0.13
hk48	5731	0.50	30	30	*	0.00	**30**	*	**0.10**	**30**	*	0.19	**30**	*	0.24
eil51	213	0.50	29	29	*	0.00	**29**	*	0.10	**29**	*	**0.06**	**29**	*	0.24
berlin52	3771	0.50	37	37	*	0.00	**37**	*	**0.09**	**37**	*	0.22	**37**	*	0.30
brazil58	12698	0.50	46	46	*	0.00	**46**	*	**0.10**	**46**	*	0.32	**46**	*	1.00
st70	338	0.50	43	43	*	0.10	**43**	*	**0.19**	**43**	*	0.29	**43**	*	0.32
eil76	269	0.50	47	47	*	0.10	46	2.13	**0.19**	**47**	*	0.20	46	2.13	0.32
pr76	54080	0.50	49	49	*	0.10	**49**	*	**0.19**	**49**	*	0.94	**49**	*	0.61
gr96	27605	0.50	64	64	*	0.10	**64**	*	**0.37**	**64**	*	1.91	**64**	*	1.44
rat99	606	0.50	52	52	*	0.40	51	1.92	**0.22**	**52**	*	0.48	**52**	*	0.66
kroA100	10641	0.50	56	56	*	0.40	**56**	*	**0.27**	**56**	*	1.33	55	1.79	0.34
kroB100	11071	0.50	58	58	*	95.40	**58**	*	**0.35**	**58**	*	1.57	57	1.72	0.63
kroC100	10375	0.50	56	56	*	0.40	**56**	*	**0.41**	**56**	*	1.12	**56**	*	0.48
kroD100	10647	0.50	59	59	*	0.10	**59**	*	**0.30**	**59**	*	1.56	58	1.69	0.65
kroE100	11034	0.50	57	57	*	159.20	55	3.51	**0.28**	**57**	*	1.42	**57**	*	0.50
rd100	3955	0.50	61	61	*	0.20	**61**	*	**0.38**	**61**	*	1.32	**61**	*	0.74
eil101	315	0.50	64	64	*	0.10	63	1.56	**0.37**	**64**	*	0.48	**64**	*	0.79
lin105	7190	0.50	66	66	*	0.30	**66**	*	**0.26**	**66**	*	2.77	**66**	*	1.42
pr107	22152	0.50	54	54	*	0.30	**54**	*	**0.21**	**54**	*	0.73	**54**	*	0.93
gr120	3471	0.50	75	75	*	0.10	74	1.33	**0.48**	**75**	*	2.29	74	1.33	1.20
pr124	29515	0.50	75	75	*	0.30	**75**	*	**0.31**	**75**	*	5.20	**75**	*	1.11
bier127	59141	0.50	103	103	*	0.30	**103**	*	**0.44**	**103**	*	3.85	**103**	*	1.18
pr136	48386	0.50	71	71	*	1.40	69	2.82	**0.51**	70	1.41	1.31	**71**	*	0.96
gr137	34927	0.50	81	81	*	1.50	**81**	*	**0.61**	**81**	*	7.10	78	3.70	3.44
pr144	29269	0.50	77	77	*	1.30	73	5.19	**0.42**	**77**	*	5.93	**77**	*	2.61
kroA150	13262	0.50	86	86	*	175.40	85	1.16	**0.90**	**86**	*	2.64	**86**	*	1.17
kroB150	13065	0.50	87	87	*	1.20	**86**	1.15	**0.94**	**86**	1.15	2.90	**86**	1.15	1.00
pr152	36841	0.50	77	77	*	1.40	76	1.30	**0.72**	**77**	*	8.78	**77**	*	3.64
u159	21040	0.50	93	93	*	3.40	82	11.83	**0.86**	**92**	1.08	5.20	**92**	1.08	1.11
rat195	1162	0.50	102	102	*	2.60	**99**	2.94	**1.01**	**99**	2.94	2.62	**99**	2.94	1.78
d198	7890	0.50	123	123	*	3.20	120	2.44	**1.46**	122	0.81	11.62	**123**	*	6.68
kroA200	14684	0.50	117	117	*	1.20	112	4.27	2.04	**117**	*	6.53	**117**	*	**1.74**
kroB200	14719	0.50	119	119	*	14.10	117	1.68	1.68	118	0.84	7.73	**119**	*	**1.66**
gr202	20080	0.50	145	145	*	12.70	140	3.45	**2.17**	**145**	*	11.84	**145**	*	6.89
ts225	63322	0.50	124	124	*	10216.30	**124**	*	1.45	**124**	*	6.08	**124**	*	**1.28**
tsp225	1958	0.50	129	129	*	94.40	117	9.30	**1.61**	126	2.33	7.76	**127**	1.55	2.29
pr226	40185	0.50	126	126	*	166.20	121	3.97	**1.20**	**126**	*	21.55	**126**	*	6.61
gr229	67301	0.50	176	176	*	0.90	174	1.14	**2.76**	174	1.14	63.24	**176**	*	8.81
gil262	1189	0.50	158	158	*	0.90	150	5.06	3.06	151	4.43	9.56	**156**	1.27	**2.84**
pr264	24568	0.50	132	132	*	21.20	**132**	*	**1.16**	**132**	*	24.77	**132**	*	5.62
a280	1290	0.50	147	147	*	13.60	133	9.52	**2.20**	**143**	2.72	11.19	**143**	2.72	3.00
pr299	24096	0.50	162	162	*	111.50	154	4.94	3.71	158	2.47	25.32	**160**	1.23	**3.12**
lin318	21015	0.50	205	205	*	22.40	194	5.37	**5.36**	200	2.44	42.09	**202**	1.46	7.15
rd400	7641	0.50	239	239	*	37.40	218	8.79	9.62	225	5.86	30.23	**234**	2.09	**6.59**
average					*	248.05		2.15	**1.14**		0.66	7.66		**0.62**	2.12

Table B.13: Generation 1, $n > 400$

				Branch-&-Cut			2-Parameter IA			GRASP with PR			EA4OP		
instance	d0	α	opt	best	gap	time	best	gap	time	best	gap	time	best	gap	time
fl417	5931	0.50	228	228	*	-	**227**	0.44	**5.67**	226	0.88	308.91	224	1.75	11.84
gr431	85707	0.50	350	350	*	139.90	343	2.00	**12.32**	346	1.14	479.25	**349**	0.29	32.84
pr439	53609	0.50	313	313	*	833.30	**310**	0.96	14.48	305	2.56	626.66	**310**	0.96	**9.92**
pcb442	25389	0.50	251	251	*	14.90	235	6.37	8.37	235	6.37	44.02	**244**	2.79	**6.93**
d493	17501	0.50	320	320	*	347.30	297	7.19	**18.88**	312	2.50	539.47	**315**	1.56	19.10
att532	13843	0.50	363	363	*	593.00	340	6.34	**16.23**	**351**	3.31	597.00	347	4.41	23.14
ali535	101170	0.50	424	.	.	.	416	1.89	**23.60**	417	1.65	793.08	**424**	*	73.03
pa561	1382	0.50	356	356	*	2103.60	340	4.49	**17.74**	330	7.30	134.50	**348**	2.25	23.18
u574	18453	0.50	354	354	*	61.40	316	10.73	**17.53**	332	6.21	205.07	**344**	2.82	17.93
rat575	3387	0.50	322	322	*	59.50	293	9.01	19.89	302	6.21	103.68	**309**	4.04	**13.76**
p654	17322	0.50	344	327	4.94	-	**344**	*	**26.68**	343	0.29	1611.23	336	2.33	28.89
d657	24456	0.50	386	386	*	715.70	344	10.88	**20.41**	367	4.92	240.25	**377**	2.33	23.24
gr666	147179	0.50	503	503	*	634.20	474	5.77	**29.49**	490	2.58	855.40	**497**	1.19	109.54
u724	20955	0.50	439	439	*	1077.10	358	18.45	35.04	415	5.47	374.33	**429**	2.28	**27.77**
rat783	4403	0.50	438	438	*	594.30	391	10.73	43.91	407	7.08	273.84	**422**	3.65	**34.59**
dsj1000	9329844	0.50	632	.	.	.	562	11.08	136.57	604	4.43	4123.41	**632**	*	**81.20**
pr1002	129523	0.50	604	604	*	-	516	14.57	118.87	558	7.62	1864.39	**572**	5.30	**45.92**
u1060	112047	0.50	627	.	.	.	577	7.97	**85.97**	607	3.19	3026.01	**627**	*	90.04
vm1084	119649	0.50	777	777	*	4927.40	726	6.56	170.98	744	4.25	6309.60	**770**	0.90	**56.29**
pcb1173	28446	0.50	633	.	.	.	590	6.79	174.26	613	3.16	2244.38	**633**	*	**60.65**
d1291	25401	0.50	684	.	.	.	**684**	*	**173.44**	670	2.05	-	646	5.56	434.87
rl1304	126474	0.50	766	.	.	.	627	18.15	228.75	694	9.40	4790.58	**766**	*	**102.45**
rl1323	135100	0.50	811	811	*	-	674	16.89	327.32	706	12.95	6345.53	**782**	3.58	**89.68**
nrw1379	28319	0.50	771	.	.	.	740	4.02	351.37	747	3.11	4934.80	**771**	*	**106.97**
fl1400	10064	0.50	1043	909	12.85	-	935	10.35	**477.51**	950	8.92	-	**1043**	*	518.25
u1432	76485	0.50	738	.	.	.	697	5.56	249.66	693	6.10	1252.50	**738**	*	**121.46**
fl1577	11125	0.50	880	.	.	.	813	7.61	442.40	837	4.89	-	**880**	*	**286.47**
d1655	31064	0.50	846	.	.	.	820	3.07	**566.47**	800	5.44	7889.11	**846**	*	757.70
vm1748	168278	0.50	1246	873	29.94	-	1195	4.09	647.09	1219	2.17	-	**1246**	*	**178.50**
u1817	28601	0.50	879	.	.	.	865	1.59	**771.89**	864	1.71	-	**879**	*	975.58
rl1889	158268	0.50	1167	890	23.74	-	1051	9.94	1291.33	1081	7.37	-	**1167**	*	**269.81**
d2103	40225	0.50	1069	.	.	.	1018	4.77	993.63	1000	6.45	-	**1069**	*	**951.27**
u2152	32127	0.50	1048	.	.	.	1021	2.58	1721.31	1019	2.77	-	**1048**	*	**1350.23**
u2319	117128	0.50	1167	.	.	.	1145	1.89	730.20	1165	0.17	3669.11	**1167**	*	**423.26**
pr2392	189016	0.50	1292	1140	11.76	-	1159	10.29	1461.82	NA	NA	-	**1292**	*	**402.29**
pcb3038	68847	0.50	1572	.	.	.	1492	5.09	1732.09	NA	NA	-	**1572**	*	**681.94**
fl3795	14386	0.50	1815	.	.	.	1776	2.15	5120.78	NA	NA	-	**1815**	*	**2994.90**
fnl4461	91283	0.50	2350	.	.	.	2245	4.47	5286.82	NA	NA	-	**2350**	*	**2462.65**
rl5915	282765	0.50	3358	.	.	.	2868	14.59	12111.74	NA	NA	-	**3358**	*	**5361.54**
rl5934	278023	0.50	3145	.	.	.	2925	7.00	9815.39	NA	NA	-	**3145**	*	**5382.25**
pla7397	11630364	0.50	5141	.	.	.	3692	28.19	**9003.06**	NA	NA	-	**5141**	*	15981.78
average					3.96	7433.41		7.43	1329.29		4.55	7893.56		**1.17**	**990.82**

Table B.14: Generation 2, $n \leq 400$

				Branch-&-Cut			2-Parameter IA			GRASP with PR			EA4OP		
instance	d0	α	opt	best	gap	time	best	gap	time	best	gap	time	best	gap	time
att48	5314	0.50	1717	1717	*	0.00	**1717**	*	**0.08**	**1717**	*	0.16	**1717**	*	0.32
gr48	2523	0.50	1761	1761	*	0.20	1750	0.62	**0.13**	**1761**	*	0.15	1749	0.68	0.20
hk48	5731	0.50	1614	1614	*	0.10	**1614**	*	**0.11**	**1614**	*	0.14	**1614**	*	0.16
eil51	213	0.50	1674	1674	*	0.40	**1674**	*	0.20	**1674**	*	**0.14**	1668	0.36	0.18
berlin52	3771	0.50	1897	1897	*	93.40	**1897**	*	**0.10**	**1897**	*	0.19	**1897**	*	0.35
brazil58	12698	0.50	2220	2220	*	0.10	**2220**	*	**0.17**	**2220**	*	0.31	2218	0.09	1.52
st70	338	0.50	2286	2286	*	19.40	2285	0.04	0.35	**2286**	*	0.35	2285	0.04	**0.31**
eil76	269	0.50	2550	2550	*	0.10	2540	0.39	**0.31**	**2550**	*	0.35	**2550**	*	0.43
pr76	54080	0.50	2708	2708	*	0.40	**2708**	*	**0.32**	**2708**	*	0.52	**2708**	*	0.48
gr96	27605	0.50	3396	3396	*	1.70	3394	0.06	**0.51**	**3396**	*	0.72	3394	0.06	1.44
rat99	606	0.50	2944	2944	*	0.90	2932	0.41	0.55	**2944**	*	**0.47**	**2944**	*	0.49
kroA100	10641	0.50	3212	3212	*	0.90	**3212**	*	0.69	**3212**	*	0.69	**3212**	*	**0.57**
kroB100	11071	0.50	3241	3241	*	6.70	3239	0.06	**0.50**	**3241**	*	0.62	3238	0.09	0.52
kroC100	10375	0.50	2947	2947	*	85.60	**2947**	*	**0.53**	2909	1.29	0.59	2931	0.54	0.60
kroD100	10647	0.50	3307	3307	*	45.00	3295	0.36	**0.56**	**3307**	*	0.72	**3307**	*	0.65
kroE100	11034	0.50	3090	3090	*	230.10	**3090**	*	0.57	3082	0.26	0.64	3082	0.26	**0.50**
rd100	3955	0.50	3359	3359	*	0.20	3351	0.24	0.52	3351	0.24	0.70	**3359**	*	**0.50**
eil101	315	0.50	3655	3655	*	153.00	3636	0.52	**0.50**	3643	0.33	0.62	**3655**	*	0.82
lin105	7190	0.50	3544	3544	*	67.30	3536	0.23	**0.56**	**3544**	*	1.12	3530	0.40	1.10
pr107	22152	0.50	2667	2667	*	0.60	**2667**	*	**0.40**	**2667**	*	0.55	**2667**	*	1.05
gr120	3471	0.50	4371	4371	*	35.80	4358	0.30	**0.74**	**4371**	*	0.84	4356	0.34	1.37
pr124	29515	0.50	3917	3917	*	0.50	**3917**	*	**0.54**	3901	0.41	1.96	3899	0.46	1.34
bier127	59141	0.50	5383	5383	*	58.80	5328	1.02	**1.08**	5331	0.97	2.12	**5381**	0.04	1.71
pr136	48386	0.50	4309	4309	*	2.10	4244	1.51	**1.02**	4228	1.88	1.03	**4309**	*	1.15
gr137	34927	0.50	4286	4286	*	196.90	**4281**	0.12	**0.87**	4270	0.37	1.79	4099	4.36	3.09
pr144	29269	0.50	4003	4003	*	90.40	3963	1.00	**0.74**	**4003**	*	2.54	3965	0.95	3.02
kroA150	13262	0.50	4918	4918	*	241.40	**4913**	0.10	1.80	4842	1.55	**1.05**	4902	0.33	1.26
kroB150	13065	0.50	4869	4869	*	24.80	4853	0.33	1.32	4853	0.33	**1.08**	**4869**	*	1.19
pr152	36841	0.50	4279	4279	*	2.20	**4269**	0.23	**1.38**	4227	1.22	3.47	4245	0.79	3.47
u159	21040	0.50	4960	4960	*	192.20	4938	0.44	1.90	4889	1.43	1.83	**4941**	0.38	**1.44**
rat195	1162	0.50	5791	5791	*	128.80	5666	2.16	1.99	5612	3.09	1.64	**5703**	1.52	**1.55**
d198	7890	0.50	6670	6670	*	74.20	6622	0.72	**2.10**	6625	0.67	4.39	**6660**	0.15	7.33
kroA200	14684	0.50	6547	6547	*	68.70	6461	1.31	2.67	6279	4.09	2.76	**6534**	0.20	**1.71**
kroB200	14719	0.50	6419	6419	*	34.70	**6328**	1.42	2.69	6282	2.13	2.16	6278	2.20	**1.97**
gr202	20080	0.50	7789	7789	*	85.70	7703	1.10	**2.13**	7659	1.67	4.99	**7789**	*	8.77
ts225	63322	0.50	6834	6834	*	6.60	6749	1.24	1.68	6743	1.33	2.75	**6819**	0.22	**1.47**
tsp225	1958	0.50	6987	6987	*	174.50	**6936**	0.73	2.60	6818	2.42	2.94	**6936**	0.73	**1.87**
pr226	40185	0.50	6662	6662	*	74.10	6646	0.24	**2.67**	6621	0.62	4.96	**6658**	0.06	7.29
gr229	67301	0.50	9177	9177	*	182.60	9111	0.72	**3.33**	9046	1.43	11.59	**9174**	0.03	13.19
gil262	1189	0.50	8321	8321	*	89.60	8100	2.66	5.20	7907	4.98	**3.15**	**8175**	1.75	3.47
pr264	24568	0.50	6654	6654	*	23.00	6244	6.16	**4.01**	**6654**	*	8.14	6173	7.23	5.94
a280	1290	0.50	8428	8428	*	103.80	8269	1.89	5.61	8021	4.83	4.60	**8304**	1.47	**2.85**
pr299	24096	0.50	9182	9182	*	426.50	9060	1.33	5.45	8846	3.66	9.35	**9112**	0.76	**3.23**
lin318	21015	0.50	10923	10923	*	862.40	10724	1.82	9.39	10424	4.57	9.77	**10866**	0.52	**8.29**
rd400	7641	0.50	13652	13652	*	293.50	13255	2.91	15.66	12617	7.58	11.09	**13442**	1.54	**6.80**
average					*	92.89		0.76	**1.92**		1.19	2.48		**0.63**	2.38

Table B.15: Generation 2, $n > 400$

				Branch-&-Cut			2-Parameter IA			GRASP with PR			EA4OP		
instance	d0	α	opt	best	gap	time	best	gap	time	best	gap	time	best	gap	time
fl417	5931	0.50	11894	11894	*	-	**11873**	0.18	**9.40**	11787	0.90	105.45	11787	0.90	16.73
gr431	85707	0.50	18318	18318	*	969.50	18112	1.12	**20.07**	17908	2.24	174.86	**18287**	0.17	51.38
pr439	53609	0.50	16171	16171	*	1298.30	15505	4.12	24.62	15698	2.92	230.64	**16085**	0.53	**11.77**
pcb442	25389	0.50	14484	14484	*	6259.10	13895	4.07	17.77	13595	6.14	15.03	**14273**	1.46	**6.83**
d493	17501	0.50	16729	.	.	.	16450	1.67	19.50	16355	2.24	103.18	**16729**	*	**17.15**
att532	13843	0.50	19598	19598	*	-	18755	4.30	36.95	18903	3.55	109.26	**19265**	1.70	**23.43**
ali535	101170	0.50	21954	21954	*	2099.70	21394	2.55	**37.79**	21202	3.43	175.15	**21910**	0.20	95.05
pa561	1382	0.50	19576	19576	*	1487.10	18279	6.63	29.36	17904	8.54	32.39	**18894**	3.48	**23.45**
u574	18453	0.50	19351	19351	*	612.50	18809	2.80	36.86	17785	8.09	36.94	**18966**	1.99	**16.33**
rat575	3387	0.50	18251	18251	*	931.10	17670	3.18	30.94	17293	5.25	32.20	**17705**	2.99	**14.97**
p654	17322	0.50	17821	17160	3.71	-	17182	3.59	**37.39**	17358	2.60	457.42	**17821**	*	42.82
d657	24456	0.50	21503	21503	*	2682.40	19969	7.13	43.01	19253	10.46	49.17	**21162**	1.59	**22.90**
gr666	147179	0.50	26336	.	.	.	26064	1.03	**52.93**	25657	2.58	351.18	**26336**	*	136.48
u724	20955	0.50	24223	24223	*	5830.50	23311	3.77	53.95	22852	5.66	69.00	**23793**	1.78	**28.71**
rat783	4403	0.50	24861	.	.	.	24098	3.07	62.44	23617	5.00	88.40	**24861**	*	**32.36**
dsj1000	9329844	0.50	35772	35772	*	-	33354	6.76	142.94	32630	8.78	409.18	**34463**	3.66	**83.34**
pr1002	129523	0.50	31746	27066	14.74	-	30440	4.11	95.47	29416	7.34	219.21	**31746**	*	**46.19**
u1060	112047	0.50	35110	.	.	.	34061	2.99	138.50	32184	8.33	367.08	**35110**	*	**77.78**
vm1084	119649	0.50	40687	40687	*	-	38642	5.03	267.85	37699	7.34	758.46	**40308**	0.93	**55.67**
pcb1173	28446	0.50	35826	.	.	.	33992	5.12	204.88	33096	7.62	370.54	**35826**	*	**69.94**
d1291	25401	0.50	35153	.	.	.	31880	9.31	304.48	33781	3.90	1735.53	**35153**	*	**289.25**
rl1304	126474	0.50	40561	.	.	.	38654	4.70	422.77	35268	13.05	732.29	**40561**	*	**97.68**
rl1323	135100	0.50	43347	43347	*	-	37905	12.55	326.45	35908	17.16	799.79	**41459**	4.36	**89.78**
nrw1379	28319	0.50	45602	.	.	.	42693	6.38	273.21	42690	6.39	694.02	**45602**	*	**117.51**
fl1400	10064	0.50	56258	53222	5.40	-	53329	5.21	**567.33**	49614	11.81	4025.94	**56258**	*	794.15
u1432	76485	0.50	44810	.	.	.	41791	6.74	248.94	41956	6.37	716.32	**44810**	*	**100.91**
fl1577	11125	0.50	45505	.	.	.	37061	18.56	510.23	44675	1.82	15494.72	**45505**	*	**334.28**
d1655	31064	0.50	47211	.	.	.	44895	4.91	**551.08**	44080	6.63	1548.97	**47211**	*	683.17
vm1748	168278	0.50	66685	.	.	.	65106	2.37	1536.33	62818	5.80	15203.11	**66685**	*	**195.85**
u1817	28601	0.50	50366	.	.	.	47606	5.48	**589.37**	47196	6.29	2025.62	**50366**	*	734.39
rl1889	158268	0.50	60084	52047	13.38	-	57552	4.21	1612.05	53798	10.46	4273.16	**60084**	*	**286.07**
d2103	40225	0.50	57202	.	.	.	50715	11.34	1038.20	53128	7.12	6370.31	**57202**	*	**682.28**
u2152	32127	0.50	60211	53976	10.36	-	56556	6.07	1184.77	55250	8.24	4063.42	**60211**	*	**1164.38**
u2319	117128	0.50	78102	72790	6.80	-	73848	5.45	1915.20	74799	4.23	3624.98	**78102**	*	**447.06**
pr2392	189016	0.50	71018	64577	9.07	-	67711	4.66	1828.25	65111	8.32	6519.53	**71018**	*	**440.57**
pcb3038	68847	0.50	91842	83951	8.59	-	85176	7.26	4158.91	85813	6.56	-	**91842**	*	**820.37**
fl3795	14386	0.50	103397	.	.	.	92086	10.94	9148.20	NA	NA	-	**103397**	*	**4788.96**
fnl4461	91283	0.50	140424	.	.	.	134030	4.55	8559.91	NA	NA	-	**140424**	*	**2618.15**
rl5915	282765	0.50	176678	.	.	.	164345	6.98	-	NA	NA	-	**176678**	*	**5512.40**
rl5934	278023	0.50	171649	.	.	.	161816	5.73	-	NA	NA	-	**171649**	*	**5757.80**
pla7397	11630364	0.50	272452	.	.	.	229543	15.75	-	NA	NA	-	**272452**	*	-
average					3.27	11644.10		5.67	2198.50		6.48	4389.82		**0.63**	**1093.37**

Table B.16: Generation 3, $n \leq 400$

instance	d0	α	opt	Branch-&-Cut best	Branch-&-Cut gap	Branch-&-Cut time	2-Parameter IA best	2-Parameter IA gap	2-Parameter IA time	GRASP with PR best	GRASP with PR gap	GRASP with PR time	EA4OP best	EA4OP gap	EA4OP time
att48	5314	0.50	1049	1049	*	38.50	**1049**	*	**0.13**	**1049**	*	0.18	**1049**	*	0.26
gr48	2523	0.50	1480	1480	*	0.20	**1480**	*	**0.07**	**1480**	*	0.20	**1480**	*	0.13
hk48	5731	0.50	1764	1764	*	0.00	**1764**	*	**0.09**	**1764**	*	0.14	**1764**	*	0.22
eil51	213	0.50	1399	1399	*	0.20	**1399**	*	**0.12**	**1399**	*	0.17	1398	0.07	0.22
berlin52	3771	0.50	1036	1036	*	124.70	**1036**	*	**0.19**	**1036**	*	0.30	1034	0.19	0.64
brazil58	12698	0.50	1702	1702	*	0.00	**1702**	*	**0.13**	**1702**	*	0.33	**1702**	*	0.71
st70	338	0.50	2108	2108	*	0.40	**2108**	*	**0.24**	**2108**	*	0.37	**2108**	*	0.31
eil76	269	0.50	2467	2467	*	0.40	2461	0.24	**0.30**	2462	0.20	0.44	**2467**	*	0.36
pr76	54080	0.50	2430	2430	*	0.20	**2430**	*	**0.26**	**2430**	*	0.56	**2430**	*	0.57
gr96	27605	0.50	3170	3170	*	61.50	**3170**	*	**0.39**	3153	0.54	1.07	3166	0.13	1.41
rat99	606	0.50	2908	2908	*	4.90	**2896**	0.41	**0.47**	2881	0.93	0.80	2886	0.76	0.78
kroA100	10641	0.50	3211	3211	*	63.30	**3211**	*	**0.30**	**3211**	*	1.16	3180	0.97	0.38
kroB100	11071	0.50	2804	2804	*	0.60	**2804**	*	**0.46**	**2804**	*	1.34	2785	0.68	0.51
kroC100	10375	0.50	3155	3155	*	1.50	**3155**	*	**0.38**	3149	0.19	0.86	**3155**	*	0.44
kroD100	10647	0.50	3167	3167	*	10.70	3123	1.39	0.65	**3167**	*	1.18	3141	0.82	**0.58**
kroE100	11034	0.50	3049	3049	*	1.50	3027	0.72	0.56	**3049**	*	1.48	**3049**	*	**0.47**
rd100	3955	0.50	2926	2926	*	113.20	**2924**	0.07	0.62	**2924**	0.07	0.90	2923	0.10	**0.48**
eil101	315	0.50	3345	3345	*	29.80	3333	0.36	**0.46**	3322	0.69	0.76	**3345**	*	0.56
lin105	7190	0.50	2986	2986	*	51.90	**2986**	*	**0.54**	**2986**	*	1.89	2973	0.44	2.09
pr107	22152	0.50	1877	1877	*	660.90	**1877**	*	**0.29**	**1877**	*	1.15	1802	4.00	0.82
gr120	3471	0.50	3779	3779	*	1.50	3736	1.14	**0.96**	3745	0.90	1.15	**3748**	0.82	1.36
pr124	29515	0.50	3557	3557	*	1021.50	3517	1.12	**0.62**	**3549**	0.22	2.41	3455	2.87	0.88
bier127	59141	0.50	2365	2365	*	79.90	2356	0.38	**1.08**	2332	1.40	2.07	**2361**	0.17	2.62
pr136	48386	0.50	4390	4390	*	86.70	**4390**	*	**0.93**	4380	0.23	2.56	**4390**	*	1.13
gr137	34927	0.50	3954	3954	*	8.60	3928	0.66	**1.13**	3926	0.71	1.89	**3954**	*	1.88
pr144	29269	0.50	3745	3745	*	112.60	3633	2.99	**0.77**	**3745**	*	3.36	3700	1.20	2.41
kroA150	13262	0.50	5039	5039	*	330.70	**5037**	0.04	1.26	5018	0.42	3.06	5019	0.40	**1.07**
kroB150	13065	0.50	5314	5314	*	107.60	5267	0.88	1.31	5272	0.79	2.31	**5314**	*	**1.04**
pr152	36841	0.50	3905	3905	*	1122.40	3557	8.91	**0.80**	**3905**	*	4.07	3902	0.08	3.62
u159	21040	0.50	5272	5272	*	52.20	**5272**	*	1.33	**5272**	*	4.46	**5272**	*	**0.94**
rat195	1162	0.50	6195	6195	*	49.90	**6174**	0.34	2.22	6086	1.76	3.06	6139	0.90	**2.00**
d198	7890	0.50	6320	6320	*	286.10	5985	5.30	**1.86**	6162	2.50	5.86	**6290**	0.47	7.14
kroA200	14684	0.50	6123	6123	*	122.30	6048	1.22	2.73	6084	0.64	4.64	**6114**	0.15	**1.72**
kroB200	14719	0.50	6266	6266	*	40.10	**6251**	0.24	2.79	6190	1.21	5.46	6213	0.85	**1.77**
gr202	20080	0.50	8616	8616	*	224.80	8111	5.86	**2.05**	8419	2.29	9.12	**8605**	0.13	10.45
ts225	63322	0.50	7575	7575	*	171.20	7149	5.62	1.47	7510	0.86	6.15	**7575**	*	**1.14**
tsp225	1958	0.50	7740	7740	*	150.30	7353	5.00	**2.38**	**7565**	2.26	5.04	7488	3.26	2.58
pr226	40185	0.50	6993	6993	*	32.60	6652	4.88	**1.97**	**6964**	0.41	15.50	6908	1.22	8.01
gr229	67301	0.50	6328	6328	*	10.20	6190	2.18	**4.42**	6205	1.94	9.03	**6297**	0.49	11.65
gil262	1189	0.50	9246	9246	*	133.40	8915	3.58	5.68	8922	3.50	6.07	**9094**	1.64	**3.94**
pr264	24568	0.50	8137	8137	*	20.70	7820	3.90	3.98	7959	2.19	17.88	**8068**	0.85	**3.62**
a280	1290	0.50	9774	9774	*	213.30	8719	10.79	4.53	**9426**	3.56	9.42	8684	11.15	**3.22**
pr299	24096	0.50	10343	10343	*	363.60	**10305**	0.37	6.07	10033	3.00	19.61	9959	3.71	**3.95**
lin318	21015	0.50	10368	10368	*	534.80	9909	4.43	7.57	9758	5.88	12.18	**10273**	0.92	**6.33**
rd400	7641	0.50	13223	13223	*	293.20	12828	2.99	14.49	12678	4.12	16.46	**13088**	1.02	**7.74**
average					*	149.66		1.69	**1.80**		0.96	4.18		**0.90**	2.31

Table B.17: Generation 3, $n > 400$

				Branch-&-Cut			2-Parameter IA			GRASP with PR			EA4OP		
instance	d0	α	opt	best	gap	time	best	gap	time	best	gap	time	best	gap	time
fl417	5931	0.50	14220	14220	*	6227.60	12792	10.04	12.49	13709	3.59	73.99	**14186**	0.24	**12.45**
gr431	85707	0.50	10911	10911	*	1046.90	10735	1.61	**17.18**	10500	3.77	103.88	**10817**	0.86	54.50
pr439	53609	0.50	15160	15160	*	-	13006	14.21	15.35	14694	3.07	153.19	**15097**	0.42	**10.96**
pcb442	25389	0.50	14819	14819	*	-	14446	2.52	11.57	14206	4.14	31.56	**14522**	2.00	**6.58**
d493	17501	0.50	25167	25167	*	-	21458	14.74	**15.15**	23362	7.17	197.43	**24981**	0.74	19.18
att532	13843	0.50	15498	15498	*	933.20	15178	2.06	23.23	14573	5.97	75.56	**15342**	1.01	**22.75**
ali535	101170	0.50	9328	.	.	.	8884	4.76	**26.04**	8672	7.03	162.88	**9328**	*	94.09
pa561	1382	0.50	14482	14482	*	10543.80	13662	5.66	35.44	13271	8.36	36.99	**14034**	3.09	**21.35**
u574	18453	0.50	20064	20064	*	1409.30	19368	3.47	34.05	18747	6.56	44.60	**19691**	1.86	**19.77**
rat575	3387	0.50	20109	20109	*	1426.50	19669	2.19	33.87	19007	5.48	47.82	**19879**	1.14	**18.03**
p654	17322	0.50	24492	24492	*	-	22303	8.94	30.94	**24221**	1.11	284.87	24130	1.48	**18.54**
d657	24456	0.50	24562	24562	*	4053.30	22401	8.80	32.94	21893	10.87	69.62	**23772**	3.22	**21.89**
gr666	147179	0.50	17020	17020	*	-	15561	8.57	**46.77**	15545	8.67	227.61	**16902**	0.69	143.87
u724	20955	0.50	28348	28348	*	5870.60	27072	4.50	58.19	26665	5.94	150.49	**27932**	1.47	**29.26**
rat783	4403	0.50	27566	27566	*	7232.30	**26870**	2.52	80.85	25591	7.16	153.06	26797	2.79	**30.64**
dsj1000	9329844	0.50	30943	.	.	.	30043	2.91	183.75	28822	6.85	781.25	**30943**	*	**79.18**
pr1002	129523	0.50	39449	39449	*	-	37244	5.59	115.38	35808	9.23	485.46	**38762**	1.74	**47.30**
u1060	112047	0.50	36570	.	.	.	35649	2.52	179.48	34873	4.64	689.68	**36570**	*	**75.88**
vm1084	119649	0.50	37653	37653	*	-	36170	3.94	167.56	36121	4.07	813.15	**37508**	0.39	**54.21**
pcb1173	28446	0.50	40069	.	.	.	38284	4.45	301.94	37506	6.40	477.29	**40069**	*	**66.16**
d1291	25401	0.50	38132	30106	21.05	-	36419	4.49	**212.23**	36063	5.43	1288.60	**38132**	*	299.87
rl1304	126474	0.50	41214	40478	1.79	-	37562	8.86	362.55	37859	8.14	1000.74	**41214**	*	**81.11**
rl1323	135100	0.50	46641	44458	4.68	-	43029	7.74	323.11	42990	7.83	904.51	**46641**	*	**93.53**
nrw1379	28319	0.50	43972	.	.	.	42412	3.55	418.50	40170	8.65	870.77	**43972**	*	**124.75**
fl1400	10064	0.50	57226	54792	4.25	-	57131	0.17	**471.99**	55269	3.42	7075.68	**57226**	*	599.81
u1432	76485	0.50	46657	.	.	.	45806	1.82	305.25	45084	3.37	1291.16	**46657**	*	**138.02**
fl1577	11125	0.50	45692	.	.	.	44188	3.29	428.11	44062	3.57	9751.32	**45692**	*	**295.62**
d1655	31064	0.50	58728	51168	12.87	-	55771	5.04	**600.91**	54121	7.84	3487.68	**58728**	*	674.25
vm1748	168278	0.50	70958	68979	2.79	-	67785	4.47	1280.00	68976	2.79	6251.95	**70958**	*	**225.29**
u1817	28601	0.50	63639	52186	18.00	-	60751	4.54	**738.77**	59783	6.06	4171.11	**63639**	*	1302.35
rl1889	158268	0.50	68422	43374	36.61	-	64660	5.50	1260.33	62538	8.60	5535.04	**68422**	*	**244.97**
d2103	40225	0.50	78084	76035	2.62	-	**78084**	*	1585.02	73034	6.47	-	77333	0.96	**1168.90**
u2152	32127	0.50	73400	52091	29.03	-	71469	2.63	**1326.63**	68152	7.15	9579.31	**73400**	*	1619.61
u2319	117128	0.50	79351	79351	*	-	**78319**	1.30	1210.42	76250	3.91	6496.30	78113	1.56	**569.76**
pr2392	189016	0.50	84094	60225	28.38	-	79704	5.22	1496.23	78364	6.81	8624.02	**84094**	*	**422.73**
pcb3038	68847	0.50	104667	96356	7.94	-	100660	3.83	4491.30	97596	6.76	-	**104667**	*	**917.39**
fl3795	14386	0.50	97707	.	.	.	95675	2.08	6867.61	NA	NA	-	**97707**	*	**3158.89**
fnl4461	91283	0.50	164201	.	.	.	158654	3.38	11047.56	NA	NA	-	**164201**	*	**3248.64**
rl5915	282765	0.50	199336	.	.	.	189096	5.14	15139.79	NA	NA	-	**199336**	*	**5593.23**
rl5934	278023	0.50	207385	.	.	.	198428	4.32	16384.22	NA	NA	-	**207385**	*	**5881.87**
pla7397	11630364	0.50	320744	.	.	.	303425	5.40	-	NA	NA	-	**320744**	*	-
average					5.86	13749.78		4.80	2082.26		6.02	4814.36		**0.63**	**1109.93**

Table B.18: Generation 4, $n \leq 400$

				Branch-&-Cut			2-Parameter IA			GRASP with PR			EA4OP		
instance	d0	α	opt	best	gap	time	best	gap	time	best	gap	time	best	gap	time
att48	6909	0.65	1870	1870	*	106.00	**1870**	*	**0.12**	**1870**	*	0.16	**1870**	*	0.52
gr48	4037	0.80	2264	2264	*	22.40	**2264**	*	**0.14**	**2264**	*	0.15	**2264**	*	0.40
hk48	9169	0.80	2177	2177	*	0.20	**2177**	*	0.14	**2177**	*	**0.13**	**2177**	*	0.15
eil51	384	0.90	2490	2490	*	82.10	2481	0.36	**0.13**	2486	0.16	0.14	**2490**	*	0.24
berlin52	4526	0.60	2089	2089	*	115.00	2085	0.19	**0.13**	**2089**	*	0.23	2085	0.19	0.48
brazil58	11428	0.45	2070	2070	*	132.00	**2070**	*	**0.18**	**2070**	*	0.29	2060	0.48	1.08
st70	574	0.85	3316	3316	*	127.70	**3314**	0.06	**0.33**	**3314**	0.06	**0.33**	**3314**	0.06	0.42
eil76	458	0.85	3646	3646	*	45.10	3638	0.22	0.37	3632	0.38	**0.33**	**3646**	*	0.52
pr76	75712	0.70	3361	3361	*	1047.70	3358	0.09	**0.31**	3358	0.09	0.52	**3361**	*	0.62
gr96	52449	0.95	4851	4851	*	212.30	**4851**	*	0.36	**4851**	*	**0.33**	**4851**	*	0.37
rat99	727	0.60	3502	3502	*	16.00	3488	0.40	0.72	3488	0.40	**0.50**	**3502**	*	0.60
kroA100	20218	0.95	4999	4999	*	187.10	**4999**	*	0.54	**4999**	*	0.60	**4999**	*	**0.36**
kroB100	9964	0.45	2935	2935	*	34.40	**2935**	*	**0.53**	**2935**	*	0.62	**2935**	*	0.61
kroC100	7263	0.35	1962	1962	*	261.60	**1962**	*	0.60	1950	0.61	**0.44**	1955	0.36	0.46
kroD100	4259	0.20	1212	1212	*	11.80	**1212**	*	0.30	**1212**	*	**0.23**	**1212**	*	0.41
kroE100	17655	0.80	4635	4635	*	203.40	4631	0.09	**0.54**	**4635**	*	0.82	4616	0.41	0.69
rd100	4747	0.60	3815	3815	*	164.60	**3815**	*	**0.57**	**3815**	*	0.76	3808	0.18	0.75
eil101	409	0.65	4308	4308	*	90.80	4294	0.32	0.70	4300	0.19	**0.59**	**4306**	0.05	0.83
lin105	5033	0.35	2455	2455	*	1020.60	**2455**	*	**0.38**	**2455**	*	1.00	2453	0.08	0.81
pr107	13291	0.30	2072	2072	*	159.00	**2072**	*	**0.35**	**2072**	*	0.92	**2072**	*	1.95
gr120	5901	0.85	5830	5830	*	236.70	5817	0.22	**0.81**	5814	0.27	1.14	**5830**	*	1.25
pr124	17710	0.30	2036	2036	*	163.80	**2036**	*	**0.37**	**2036**	*	0.78	1937	4.86	1.18
bier127	53227	0.45	5068	5068	*	278.40	5045	0.45	**1.19**	5046	0.43	2.28	**5067**	0.02	2.28
pr136	33871	0.35	2860	2860	*	6303.60	2831	1.01	0.87	**2833**	0.94	0.90	2820	1.40	**0.74**
gr137	55883	0.80	6523	6523	*	203.10	6513	0.15	**1.04**	6500	0.35	2.17	**6516**	0.11	2.52
pr144	40976	0.70	5641	5641	*	357.90	5611	0.53	**0.95**	5624	0.30	2.71	**5639**	0.04	4.53
kroA150	19893	0.75	6858	6858	*	415.90	6835	0.34	**1.28**	6828	0.44	2.17	**6855**	0.04	1.69
kroB150	20904	0.80	7023	7023	*	303.00	6987	0.51	1.50	7011	0.17	2.20	**7020**	0.04	**1.16**
pr152	51578	0.70	5823	5823	*	483.60	5201	10.68	**1.30**	5819	0.07	2.64	**5820**	0.05	5.21
u159	14729	0.35	3147	3147	*	1145.20	**3147**	*	1.41	**3147**	*	1.81	**3147**	*	**0.92**
rat195	2207	0.95	9753	9753	*	205.40	9724	0.30	3.73	9630	1.26	2.36	**9750**	0.03	**1.69**
d198	6312	0.40	4661	4661	*	492.70	4589	1.54	**1.91**	4642	0.41	3.40	**4654**	0.15	4.95
kroA200	26432	0.90	9892	9892	*	340.30	9829	0.64	**2.52**	9862	0.30	4.89	**9892**	*	2.73
kroB200	26494	0.90	9849	9849	*	253.20	9796	0.54	2.50	9796	0.54	4.33	**9842**	0.07	**1.62**
gr202	6025	0.15	1071	1071	*	376.10	**1071**	*	0.63	**1071**	*	**0.47**	995	7.10	1.47
ts225	113979	0.90	11002	11002	*	3524.60	10827	1.59	2.74	10885	1.06	3.90	**11002**	*	**1.87**
tsp225	3525	0.90	10972	10972	*	706.70	10952	0.18	4.70	10912	0.55	4.40	**10972**	*	**2.52**
pr226	32148	0.40	4893	4893	*	1183.10	4868	0.51	**2.16**	4879	0.29	5.81	**4890**	0.06	4.83
gr229	121142	0.90	11482	11482	*	563.10	11451	0.27	4.79	11430	0.45	**4.02**	**11482**	*	6.46
gil262	357	0.15	2031	2031	*	1770.50	2029	0.10	1.88	**2031**	*	2.88	2030	0.05	**1.35**
pr264	34395	0.70	10253	10253	*	277.50	9808	4.34	**4.13**	9858	3.85	13.97	**10166**	0.85	6.42
a280	1935	0.75	12064	12064	*	351.80	11810	2.11	5.83	11731	2.76	7.12	**12048**	0.13	**3.39**
pr299	45782	0.95	14986	14986	*	7771.90	14894	0.61	6.71	14898	0.59	12.67	**14980**	0.04	**3.46**
lin318	35725	0.85	15132	15132	*	-	15049	0.55	**7.52**	15012	0.79	15.57	**15119**	0.09	7.91
rd400	14517	0.95	20107	20107	*	5093.10	20004	0.51	12.42	19973	0.67	19.62	**20101**	0.03	**9.61**
average					*	1218.69		0.65	**1.83**		0.41	2.96		**0.38**	2.09

Table B.19: Generation 4, $n > 400$

				Branch-&-Cut			2-Parameter IA			GRASP with PR			EA4OP		
instance	d0	α	opt	best	gap	time	best	gap	time	best	gap	time	best	gap	time
fl417	10082	0.85	20496	20496	*	-	20438	0.28	**11.82**	20366	0.63	82.41	**20494**	0.01	39.61
gr431	51425	0.30	13976	13976	*	-	13652	2.32	**18.13**	13499	3.41	165.40	**13969**	0.05	50.29
pr439	75052	0.70	19613	19613	*	3936.10	19435	0.91	23.59	19139	2.42	177.81	**19510**	0.53	**13.61**
pcb442	10156	0.20	5839	5839	*	-	**5749**	1.54	9.47	5600	4.09	9.50	5650	3.24	**3.40**
d493	24502	0.70	21740	21740	*	-	21357	1.76	28.72	20856	4.07	74.96	**21674**	0.30	**21.00**
att532	26302	0.95	26728	26728	*	-	26570	0.59	19.12	26526	0.76	73.04	**26728**	*	**17.20**
ali535	40468	0.20	13520	13520	*	15739.60	13393	0.94	**29.43**	13102	3.09	406.85	**13442**	0.58	73.07
pa561	2487	0.90	27719	27712	0.03	-	27228	1.77	31.85	26822	3.24	40.43	**27719**	*	**24.14**
u574	35060	0.95	28823	28823	*	-	28545	0.96	**24.81**	28446	1.31	70.59	**28822**	0.00	26.03
rat575	6096	0.90	28364	28364	*	-	27995	1.30	43.23	27664	2.47	39.55	**28334**	0.11	**24.68**
p654	27715	0.80	31814	31814	*	-	31657	0.49	**40.11**	31383	1.35	514.38	**31717**	0.30	123.82
d657	44021	0.90	32548	32548	*	13485.10	32059	1.50	52.01	31927	1.91	123.96	**32534**	0.04	**33.00**
gr666	103026	0.35	21013	21013	*	-	20369	3.06	**39.13**	20412	2.86	511.53	**20901**	0.53	132.65
u724	35624	0.85	34988	34988	*	-	34169	2.34	57.31	33923	3.04	108.86	**34921**	0.19	**40.93**
rat783	1321	0.15	7829	7829	*	-	7459	4.73	23.26	7203	8.00	32.23	**7548**	3.59	**13.35**
dsj1000	6530891	0.35	27357	27357	*	-	24840	9.20	127.16	25113	8.20	573.23	**25352**	7.33	**48.13**
pr1002	90666	0.35	23527	23527	*	-	21029	10.62	180.07	20224	14.04	127.34	**22482**	4.44	**35.67**
u1060	190480	0.85	51775	51768	0.01	-	50921	1.65	190.31	50302	2.85	430.51	**51775**	*	**150.58**
vm1084	107684	0.45	38678	38678	*	-	36455	5.75	201.55	35535	8.13	551.57	**38228**	1.16	**50.34**
pcb1173	48359	0.85	56010	55954	0.10	-	53687	4.15	206.44	52559	6.16	469.21	**56010**	*	**77.73**
d1291	5081	0.10	4029	4029	*	2335.60	**4029**	*	**35.22**	**4029**	*	71.53	4024	0.12	45.07
rl1304	189711	0.75	57782	57782	*	-	53775	6.93	470.54	51258	11.29	1163.85	**57545**	0.41	**112.18**
rl1323	243180	0.90	65664	65476	0.29	-	63666	3.04	492.15	61632	6.14	1070.12	**65664**	*	**99.81**
nrw1379	53807	0.95	69214	69119	0.14	-	68510	1.02	385.51	68046	1.69	974.63	**69214**	*	**152.00**
fl1400	18115	0.90	70488	70476	0.02	-	70444	0.06	**177.57**	70449	0.06	4573.17	**70488**	*	287.75
u1432	91783	0.60	54540	54540	*	-	49738	8.80	565.02	50150	8.05	910.86	**53550**	1.82	**127.79**
fl1577	7788	0.35	33754	22191	34.26	-	25157	25.47	327.70	31740	5.97	10432.87	**33754**	*	**200.71**
d1655	21745	0.35	31880	29920	6.15	-	30024	5.82	**300.33**	30040	5.77	781.57	**31880**	*	371.31
vm1748	252417	0.75	82126	81778	0.42	-	80623	1.83	776.22	79540	3.15	5491.77	**82126**	*	**265.55**
u1817	20021	0.35	36416	31800	12.68	-	33428	8.21	845.38	32142	11.74	891.10	**36416**	*	**418.80**
rl1889	237402	0.75	83081	71527	13.91	-	80240	3.42	1536.30	76078	8.43	4799.95	**83081**	*	**363.35**
d2103	24136	0.30	34192	31045	9.20	-	29811	12.81	805.24	31176	8.82	1291.98	**34192**	*	**465.36**
u2152	28914	0.45	54744	48472	11.46	-	50467	7.81	1587.17	49688	9.24	3227.28	**54744**	*	**906.84**
u2319	187405	0.80	110995	110995	*	-	108463	2.28	1218.39	107764	2.91	4687.71	**110960**	0.03	**438.26**
pr2392	132312	0.35	50902	45407	10.80	-	47791	6.11	1355.53	45506	10.60	3173.63	**50902**	*	**285.26**
pcb3038	75732	0.55	101173	91831	9.23	-	93070	8.01	2974.79	94607	6.49	-	**101173**	*	**800.13**
fl3795	11509	0.40	80069	71328	10.92	-	67850	15.26	7074.26	NA	NA	-	**80069**	*	**4496.09**
fnl4461	54770	0.30	85088	84098	1.16	-	79110	7.03	5216.83	NA	NA	-	**85088**	*	**1490.80**
rl5915	480701	0.85	279277	279116	0.06	-	264269	5.37	-	NA	NA	-	**279277**	*	**8438.60**
rl5934	222418	0.40	137838	.	.	.	128287	6.93	-	NA	NA	-	**137838**	*	**4037.07**
pla7397	5815182	0.25	142399	106131	25.47	-	121860	14.42	-	NA	NA	-	**142399**	*	**6667.36**
average					3.66	17087.41		5.04	1987.85		5.07	3807.94		**0.60**	**767.54**

B.2 Chapter 3: Shrinking and exact SEC for Cycle Problems

In this section, we show the computational results obtained in each considered SEC instance. For each instance, we present three tables: two are related with the shrinking processes and one is related with separation and SEC generation processes. In addition, the results are separated into three groups (Gen1, Gen2 and Gen3). These groups represent the generation strategy proposed in [Fischetti et al., 1998] to build the OP vertex scores which are then used to obtain the support graphs.

From Table B.20, Table B.22, ... and Table B.34, we report the details of the shrinking preprocess. One can see, below the support graph and shrunk graph columns, the size of the given support graph and the size of the shrunk support graph for each shrinking strategy. In the preprocess columns, we show the number of Q sets obtained and the time (in milliseconds) needed by each shrinking preprocess. As can be seen, the shrinking is very fast, needing very few dozens of millisecond to be accomplished in the larger instances. An interesting point of these tables is that within the shrinking preprocess we are already able to obtain Q sets that correspond with violated SECs. In particular, the largest amount of Q sets are obtained with the shrinking strategy S1S2.

In tables Table B.21, Table B.23, ... and Table B.35, we report the number of times a rule is applied by each shrinking strategy. Regarding the Conjecture 1 in the discussion of the computational experiments of Chapter 3, it can be seen that Rule C3 is rarely applied in the shrinking preprocess. Moreover, the strategy C1C2C3 does not provide further contractions of the support graph and, in all the compared instances, the obtained final shrunk graphs have the same amount of vertices and edges as with strategy C1C2.

The extra column in these tables represents how many extra vertices are contracted in the internal shrinking process of Algorithm DHI, i.e, Extra is increased by one if rule C1, C2 or S1 is applied and by two if rule C3 is applied. The results show that this extra shrinking is rarely achieved.

In tables Table B.36, Table B.37, ... and Table B.41 ,we report the details about the separation process and SEC generation. We can see that EPG approach always obtains more violated SECs than Algorithm EH as suggested theoretically in Chapter 3. Moreover, without using the shrinking preprocess, the EPG algorithm is always faster than Algorithm EH except for the smallest instance pr76.

Regarding the SEC generation process, we compare two strategies 1×1 and 10×10, which refer to the amount of vertices considered inside and outside Q sets when generating the violated SECs. What we see is that, in medium-sized instances, the generation of violated SECs is the most time-consuming part (see the results regarding Algorithm EPG), but in large-sized, this difference is shortened. Nevertheless, it is likely that most of the generated violated cuts by 10×10 (around half a million of different violated SECs were obtained in large-sized instances by EPG) are useless and counterproductive to consider them, in practice, for a B&C.

Table B.20: Graph sizes, number of obtained Q sets and running time of the preprocess by shrinking strategy and OP instance generation in pr76.

		Gen1						Gen2						Gen3					
		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess	
Shrinking	$\|V\|$	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time
NO	76	65	71	65	71	0	0.05	50	59	50	59	0	0.06	54	63	54	63	0	0.05
C1	76	65	71	26	32	0	0.10	50	59	20	29	0	0.08	54	63	24	33	0	0.09
C1C2	76	65	71	26	32	0	0.09	50	59	20	29	0	0.08	54	63	24	33	0	0.09
C1C2C3	76	65	71	26	32	0	0.10	50	59	20	29	0	0.09	54	63	24	33	0	0.10
S1	76	65	71	14	15	4	0.10	50	59	9	13	0	0.08	54	63	13	18	0	0.09
S1S2	76	65	71	11	12	4	0.10	50	59	9	13	0	0.08	54	63	13	18	0	0.09

Table B.21: Number of applications of each rule in the preprocess by shrinking strategy and OP instance generation in pr76. The extra column is particular of DHI separation strategy (during separation).

	Gen1							Gen2							Gen3						
	Preprocess						DHI	Preprocess						DHI	Preprocess						DHI
Shrinking	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra
NO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1	39	0	0	0	0	75	0	30	0	0	0	0	53	0	30	0	0	0	0	65	0
C1C2	39	0	0	0	0	75	0	30	0	0	0	0	53	0	30	0	0	0	0	65	0
C1C2C3	39	0	0	0	0	75	0	30	0	0	0	0	53	0	30	0	0	0	0	65	0
S1	0	0	0	51	0	73	0	0	0	0	41	0	62	0	0	0	0	41	0	63	0
S1S2	0	0	0	53	1	72	0	0	0	0	41	0	62	0	0	0	0	41	0	63	0

Table B.22: Graph sizes, number of obtained Q sets and running time of the preprocess by shrinking strategy and OP instance generation in att532.

		Gen1						Gen2						Gen3					
		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess	
Shrinking	$\|V\|$	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time
NO	532	458	528	458	528	0	0.33	413	503	413	503	0	0.29	412	512	412	512	0	0.32
C1	532	458	528	166	236	0	0.62	413	503	212	302	0	0.51	412	512	240	340	0	0.50
C1C2	532	458	528	142	196	3	0.67	413	503	200	279	4	0.53	412	512	221	305	6	0.54
C1C2C3	532	458	528	142	196	4	0.70	413	503	200	279	4	0.59	412	512	221	305	6	0.58
S1	532	458	528	77	111	15	0.74	413	503	119	164	29	0.63	412	512	135	185	24	0.59
S1S2	532	458	528	73	105	19	0.69	413	503	109	148	31	0.65	412	512	129	179	25	0.63

Table B.23: Number of applications of each rule in the preprocess by shrinking strategy and OP instance generation in att532. The extra column is particular of DHI separation strategy (during separation).

	Gen1							Gen2							Gen3						
	Preprocess						DHI	Preprocess						DHI	Preprocess						DHI
Shrinking	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra
NO	0	0	0	0	0	0	0.0	0	0	0	0	0	0	0.0	0	0	0	0	0	0	0.0
C1	292	0	0	0	0	531	0.0	201	0	0	0	0	462	0.0	172	0	0	0	0	466	0.0
C1C2	302	14	0	0	0	534	0.0	205	8	0	0	0	469	0.0	178	13	0	0	0	470	0.0
C1C2C3	302	10	2	0	0	528	0.0	205	8	0	0	0	469	0.0	178	13	0	0	0	470	0.0
S1	0	0	0	381	0	513	0.0	0	0	0	294	0	470	0.0	0	0	0	277	0	467	0.0
S1S2	0	0	0	381	4	508	0.5	0	0	0	296	8	469	0.0	0	0	0	278	5	470	0.2

Table B.24: Graph sizes, number of obtained Q sets and running time of the preprocess by shrinking strategy and OP instance generation in vm1084.

		Gen1						Gen2						Gen3					
		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess	
Shrinking	$\lvert V\rvert$	$\lvert V^*\rvert$	$\lvert E^*\rvert$	$\lvert V^*\rvert$	$\lvert E^*\rvert$	#Q	Time	$\lvert V^*\rvert$	$\lvert E^*\rvert$	$\lvert V^*\rvert$	$\lvert E^*\rvert$	#Q	Time	$\lvert V^*\rvert$	$\lvert E^*\rvert$	$\lvert V^*\rvert$	$\lvert E^*\rvert$	#Q	Time
NO	1084	861	980	861	980	0	0.66	863	1012	863	1012	0	0.68	785	917	785	917	0	0.63
C1	1084	861	980	297	416	0	1.16	863	1012	377	526	0	1.18	785	917	297	429	0	1.08
C1C2	1084	861	980	260	354	8	1.20	863	1012	341	465	7	1.17	785	917	267	378	7	1.11
C1C2C3	1084	861	980	260	354	8	1.30	863	1012	341	465	6	1.27	785	917	267	378	7	1.22
S1	1084	861	980	147	202	40	1.30	863	1012	213	289	45	1.30	785	917	160	233	34	1.14
S1S2	1084	861	980	134	180	48	1.39	863	1012	200	272	53	1.37	785	917	146	210	42	1.26

Table B.25: Number of applications of each rule in the preprocess by shrinking strategy and OP instance generation in vm1084. The extra column is particular of DHI separation strategy (during separation).

	Gen1							Gen2							Gen3						
	Preprocess						DHI	Preprocess						DHI	Preprocess						DHI
Shrinking	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra
NO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1	564	0	0	0	0	945	0	486	0	0	0	0	970	0	488	0	0	0	0	881	0
C1C2	582	19	0	0	0	980	0	502	20	0	0	0	971	0	500	18	0	0	0	887	0
C1C2C3	582	19	0	0	0	980	0	502	18	1	0	0	968	0	500	16	1	0	0	882	0
S1	0	0	0	714	0	962	0	0	0	0	650	0	975	0	0	0	0	625	0	876	0
S1S2	0	0	0	716	11	950	0	0	0	0	654	9	964	0	0	0	0	627	12	870	0

Table B.26: Graph sizes, number of obtained Q sets and running time of the preprocess by shrinking strategy and OP instance generation in rl1323.

		Gen1						Gen2						Gen3					
		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess	
Shrinking	$\|V\|$	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time
NO	1323	1011	1165	1011	1165	0	0.83	933	1073	933	1073	0	0.76	956	1124	956	1124	0	0.77
C1	1323	1011	1165	421	575	0	1.39	933	1073	375	515	0	1.32	956	1124	406	574	0	1.32
C1C2	1323	1011	1165	401	538	10	1.44	933	1073	335	445	12	1.36	956	1124	382	529	9	1.35
C1C2C3	1323	1011	1165	401	538	10	1.48	933	1073	335	445	12	1.46	956	1124	382	529	9	1.50
S1	1323	1011	1165	248	331	46	1.58	933	1073	209	276	59	1.46	956	1124	225	303	56	1.53
S1S2	1323	1011	1165	237	317	50	1.58	933	1073	200	262	66	1.50	956	1124	194	257	83	1.59

Table B.27: Number of applications of each rule in the preprocess by shrinking strategy and OP instance generation in rl1323. The extra column is particular of DHI separation strategy (during separation).

	Gen1							Gen2							Gen3						
	Preprocess						DHI	Preprocess						DHI	Preprocess						DHI
Shrinking	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra
NO	0	0	0	0	0	0	0.0	0	0	0	0	0	0	0.0	0	0	0	0	0	0	0.0
C1	590	0	0	0	0	1149	0.0	558	0	0	0	0	1045	0.0	550	0	0	0	0	1093	0.0
C1C2	598	12	0	0	0	1141	0.0	579	19	0	0	0	1048	0.0	559	15	0	0	0	1091	0.0
C1C2C3	598	12	0	0	0	1141	0.0	578	18	1	0	0	1044	0.0	559	13	1	0	0	1086	0.0
S1	0	0	0	763	0	1148	0.0	0	0	0	724	0	1055	0.0	0	0	0	731	0	1092	0.0
S1S2	0	0	0	764	10	1141	0.0	0	0	0	726	7	1050	0.5	0	0	0	738	24	1069	0.0

Table B.28: Graph sizes, number of obtained Q sets and running time of the preprocess by shrinking strategy and OP instance generation in vm1748.

		Gen1						Gen2						Gen3					
		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess	
Shrinking	$\|V\|$	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time
NO	1748	1490	1756	1490	1756	0	1.32	1487	1837	1487	1837	0	1.36	1361	1586	1361	1586	0	1.21
C1	1748	1490	1756	642	908	0	2.39	1487	1837	808	1158	0	2.33	1361	1586	515	740	0	2.18
C1C2	1748	1490	1756	596	823	18	2.49	1487	1837	727	1005	32	2.47	1361	1586	480	680	6	2.28
C1C2C3	1748	1490	1756	596	823	18	2.72	1487	1837	727	1005	32	2.64	1361	1586	480	680	6	2.42
S1	1748	1490	1756	374	513	76	2.87	1487	1837	487	675	106	2.88	1361	1586	284	411	48	2.51
S1S2	1748	1490	1756	337	462	87	2.86	1487	1837	455	630	121	2.81	1361	1586	249	358	72	2.63

Table B.29: Number of applications of each rule in the preprocess by shrinking strategy and OP instance generation in vm1748. The extra column is particular of DHI separation strategy (during separation).

	Gen1							Gen2							Gen3						
	Preprocess						DHI	Preprocess						DHI	Preprocess						DHI
Shrinking	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra
NO	0	0	0	0	0	0	0.0	0	0	0	0	0	0	0.0	0	0	0	0	0	0	0.0
C1	848	0	0	0	0	1653	0.0	679	0	0	0	0	1659	0.0	846	0	0	0	0	1518	0.0
C1C2	866	28	0	0	0	1676	0.0	711	49	0	0	0	1690	0.0	859	22	0	0	0	1533	0.0
C1C2C3	866	28	0	0	0	1676	0.0	711	43	3	0	0	1680	0.0	859	20	1	0	0	1529	0.0
S1	0	0	0	1116	0	1692	0.0	0	0	0	1000	0	1729	0.0	0	0	0	1077	0	1525	0.0
S1S2	0	0	0	1123	30	1684	0.0	0	0	0	1003	29	1720	0.2	0	0	0	1081	31	1501	0.8

Table B.30: Graph sizes, number of obtained Q sets and running time of the preprocess by shrinking strategy and OP instance generation in rl5934.

		Gen1						Gen2						Gen3					
		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess	
Shrinking	$\|V\|$	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time
NO	5934	4303	4871	4303	4871	0	5.67	4101	4651	4101	4651	0	5.31	3970	4424	3970	4424	0	5.43
C1	5934	4303	4871	1477	2045	0	11.10	4101	4651	1533	2083	0	10.06	3970	4424	1127	1581	0	10.44
C1C2	5934	4303	4871	1312	1754	54	11.50	4101	4651	1415	1873	44	10.45	3970	4424	1014	1381	42	10.60
C1C2C3	5934	4303	4871	1312	1754	54	11.61	4101	4651	1415	1873	43	10.71	3970	4424	1014	1381	41	10.57
S1	5934	4303	4871	800	1067	255	12.32	4101	4651	877	1123	266	11.64	3970	4424	624	848	189	10.99
S1S2	5934	4303	4871	750	990	300	12.57	4101	4651	800	1026	296	11.83	3970	4424	560	756	237	11.25

Table B.31: Number of applications of each rule in the preprocess by shrinking strategy and OP instance generation in rl5934. The extra column is particular of DHI separation strategy (during separation).

	Gen1							Gen2							Gen3						
	Preprocess						DHI	Preprocess						DHI	Preprocess						DHI
Shrinking	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra
NO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1	2826	0	0	0	0	4864	0	2568	0	0	0	0	4598	0	2843	0	0	0	0	4389	0
C1C2	2904	87	0	0	0	4854	0	2622	64	0	0	0	4603	0	2897	59	0	0	0	4374	0
C1C2C3	2904	85	1	0	0	4852	0	2620	58	4	0	0	4585	0	2896	54	3	0	0	4363	0
S1	0	0	0	3503	0	4790	0	0	0	0	3224	0	4563	0	0	0	0	3346	0	4361	0
S1S2	0	0	0	3511	42	4757	0	0	0	0	3247	54	4542	0	0	0	0	3355	55	4324	0

Table B.32: Graph sizes, number of obtained Q sets and running time of the preprocess by shrinking strategy and OP instance generation in usa13509.

		Gen1						Gen2						Gen3					
		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess	
Shrinking	$\|V\|$	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time
NO	13509	9084	9990	9084	9990	0	18.58	8015	8735	8015	8735	0	14.58	7245	7992	7245	7992	0	12.73
C1	13509	9084	9990	2017	2923	0	44.98	8015	8735	1723	2443	0	35.07	7245	7992	1644	2391	0	28.85
C1C2	13509	9084	9990	1891	2725	1	44.86	8015	8735	1623	2284	1	35.70	7245	7992	1490	2148	0	29.28
C1C2C3	13509	9084	9990	1891	2725	1	45.12	8015	8735	1623	2284	1	35.42	7245	7992	1490	2148	0	30.03
S1	13509	9084	9990	882	1347	213	49.39	8015	8735	846	1249	196	39.33	7245	7992	718	1107	146	32.58
S1S2	13509	9084	9990	705	1046	381	50.35	8015	8735	717	1027	349	39.16	7245	7992	587	885	255	32.45

Table B.33: Number of applications of each rule in the preprocess by shrinking strategy and OP instance generation in usa13509. The extra column is particular of DHI separation strategy (during separation).

	Gen1							Gen2							Gen3						
	Preprocess						DHI	Preprocess						DHI	Preprocess						DHI
Shrinking	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra
NO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1	7067	0	0	0	0	10005	0	6292	0	0	0	0	8751	0	5601	0	0	0	0	7971	0
C1C2	7123	70	0	0	0	9969	0	6334	58	0	0	0	8722	0	5666	89	0	0	0	7936	0
C1C2C3	7123	64	3	0	0	9960	0	6334	56	1	0	0	8718	0	5666	83	3	0	0	7926	0
S1	0	0	0	8202	0	9753	0	0	0	0	7169	0	8628	0	0	0	0	6527	0	7771	0
S1S2	0	0	0	8223	156	9610	0	0	0	0	7179	119	8503	0	0	0	0	6541	117	7670	0

Table B.34: Graph sizes, number of obtained Q sets and running time of the preprocess by shrinking strategy and OP instance generation in d15112.

		Gen1						Gen2						Gen3					
		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess		Support graph		Shrunk graph		Preprocess	
Shrinking	$\|V\|$	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time	$\|V^*\|$	$\|E^*\|$	$\|V^*\|$	$\|E^*\|$	#Q	Time
NO	15112	9075	9866	9075	9866	0	18.73	7682	8322	7682	8322	0	14.25	9393	10378	9393	10378	0	19.63
C1	15112	9075	9866	1793	2584	0	43.70	7682	8322	1600	2240	0	32.45	9393	10378	2170	3155	0	46.91
C1C2	15112	9075	9866	1656	2373	0	44.42	7682	8322	1510	2097	1	32.28	9393	10378	1994	2876	5	47.70
C1C2C3	15112	9075	9866	1656	2373	0	45.28	7682	8322	1510	2097	1	32.70	9393	10378	1994	2876	5	48.93
S1	15112	9075	9866	785	1203	176	48.52	7682	8322	809	1164	197	34.90	9393	10378	941	1460	163	53.01
S1S2	15112	9075	9866	658	977	305	48.12	7682	8322	690	964	297	36.06	9393	10378	796	1213	285	53.44

Table B.35: Number of applications of each rule in the preprocess by shrinking strategy and OP instance generation in d15112. The extra column is particular of DHI separation strategy (during separation).

	Gen1							Gen2							Gen3						
	Preprocess						DHI	Preprocess						DHI	Preprocess						DHI
Shrinking	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra	C1	C2	C3	S1	S2	H	Extra
NO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C1	7282	0	0	0	0	9933	0	6082	0	0	0	0	8382	0	7223	0	0	0	0	10350	0
C1C2	7345	74	0	0	0	9864	0	6120	52	0	0	0	8357	0	7299	100	0	0	0	10305	0
C1C2C3	7345	70	2	0	0	9858	0	6120	52	0	0	0	8357	0	7299	98	1	0	0	10302	0
S1	0	0	0	8290	0	9693	0	0	0	0	6873	0	8270	0	0	0	0	8452	0	10118	0
S1S2	0	0	0	8301	116	9586	0	0	0	0	6893	99	8165	0	0	0	0	8468	129	10025	0

Table B.36: Number of obtained Q sets in separation, number of generated SECs when $k_{in} \times k_{out}$ is set to 1×1 and 10×10 and their running times by separation strategy, shrinking strategy and OP instance generation in pr76.

		Gen1						Gen2						Gen3					
		Separation		SEC Generation				Separation		SEC Generation				Separation		SEC Generation			
		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)	
Sep.	Shrinking	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time
EH	NO	46	0.4	46	0.5	3030	1.5	16	0.6	16	0.7	860	1.1	37	0.3	37	0.4	2750	1.4
	C1	11	0.2	11	0.2	610	0.5	10	0.2	10	0.2	390	0.5	10	0.2	10	0.2	620	0.4
	C1C2	11	0.2	11	0.2	610	0.5	10	0.2	10	0.2	390	0.5	10	0.2	10	0.2	620	0.5
	C1C2C3	11	0.2	11	0.2	610	0.5	10	0.2	10	0.3	390	0.4	10	0.2	10	0.3	620	0.5
	S1	8	0.1	8	0.2	420	0.3	2	0.1	2	0.1	60	0.1	6	0.1	6	0.1	290	0.3
	S1S2	6	0.1	6	0.2	340	0.3	2	0.1	2	0.1	60	0.2	6	0.1	6	0.1	290	0.2
DH	NO	9	0.2	9	0.2	220	0.3	8	0.3	8	0.3	170	0.4	11	0.2	11	0.2	510	0.4
	C1	6	0.2	5	0.2	220	0.3	7	0.2	7	0.2	230	0.2	6	0.2	6	0.2	190	0.3
	C1C2	6	0.2	5	0.2	220	0.3	7	0.2	7	0.2	230	0.3	6	0.1	6	0.2	190	0.2
	C1C2C3	6	0.2	5	0.2	220	0.3	7	0.2	7	0.2	230	0.2	6	0.2	6	0.2	190	0.3
	S1	7	0.1	7	0.2	390	0.3	3	0.1	3	0.1	80	0.1	4	0.1	4	0.1	110	0.2
	S1S2	7	0.1	7	0.1	370	0.3	3	0.1	3	0.1	80	0.2	4	0.1	4	0.1	110	0.2
DHI	NO	9	0.2	9	0.2	220	0.3	8	0.3	8	0.3	170	0.4	11	0.2	11	0.2	510	0.4
	C1	6	0.2	5	0.2	220	0.3	7	0.2	7	0.2	230	0.3	6	0.2	6	0.2	190	0.3
	C1C2	6	0.2	5	0.2	220	0.3	7	0.2	7	0.2	230	0.3	6	0.2	6	0.2	190	0.3
	C1C2C3	6	0.2	5	0.2	220	0.3	7	0.2	7	0.2	230	0.3	6	0.2	6	0.2	190	0.3
	S1	7	0.1	7	0.2	390	0.3	3	0.1	3	0.1	80	0.1	4	0.1	4	0.1	110	0.2
	S1S2	7	0.1	7	0.1	370	0.3	3	0.1	3	0.1	80	0.1	4	0.1	4	0.2	110	0.2
EPG	NO	48	0.6	48	0.7	3090	2.0	20	0.8	20	0.8	960	1.2	39	0.5	39	0.6	2910	1.4
	C1	14	0.4	14	0.4	680	0.7	10	0.3	10	0.3	390	0.5	12	0.3	12	0.4	620	0.6
	C1C2	14	0.3	14	0.4	680	0.6	10	0.3	10	0.3	390	0.5	12	0.3	12	0.4	620	0.6
	C1C2C3	14	0.4	14	0.4	680	0.6	10	0.3	10	0.4	390	0.5	12	0.3	12	0.4	620	0.5
	S1	10	0.2	10	0.3	540	0.4	3	0.2	3	0.2	80	0.2	6	0.2	6	0.3	370	0.3
	S1S2	7	0.2	7	0.3	440	0.3	3	0.2	3	0.2	80	0.2	6	0.2	6	0.2	370	0.3

Table B.37: Number of obtained Q sets in separation, number of generated SECs when $k_{in} \times k_{out}$ is set to 1×1 and 10×10 and their running times by separation strategy, shrinking strategy and OP instance generation in att532.

		Gen1						Gen2						Gen3					
		Separation		SEC Generation				Separation		SEC Generation				Separation		SEC Generation			
		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)	
Sep.	Shrinking	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time
EH	NO	94	65.3	94	67.8	8550	67.4	32	54.2	32	53.9	2630	56.0	30	65.6	30	65.9	2750	67.4
	C1	43	5.4	43	6.1	3560	7.8	17	17.0	17	17.4	1470	17.6	21	20.4	21	20.6	1850	21.7
	C1C2	40	4.4	40	5.3	3250	6.0	21	14.1	21	14.1	1570	15.2	27	17.0	27	17.2	2080	18.3
	C1C2C3	41	4.9	41	5.4	3350	7.3	21	14.6	21	14.9	1570	15.2	27	16.7	27	17.5	2080	17.5
	S1	27	3.6	27	4.2	1820	4.3	40	7.2	40	8.0	2490	8.4	38	7.6	38	8.1	2160	8.8
	S1S2	31	3.2	31	3.7	2220	4.2	48	5.3	48	5.9	3240	7.2	39	6.7	39	6.6	2190	8.5
DH	NO	52	5.6	51	6.1	3280	7.8	56	8.2	55	8.6	2500	9.5	54	11.3	54	11.5	2420	12.7
	C1	34	2.2	34	2.4	1950	3.8	41	3.9	40	4.0	1940	5.4	44	5.6	44	6.1	2140	6.8
	C1C2	34	1.9	34	2.4	2060	3.1	44	3.4	43	4.1	2110	4.5	46	4.8	46	5.2	2320	6.3
	C1C2C3	35	1.9	35	2.0	2160	3.5	44	3.7	43	4.0	2110	5.5	46	4.5	46	4.8	2320	6.1
	S1	29	1.6	29	1.8	1970	2.8	59	1.9	58	2.5	3290	3.7	53	2.4	53	2.6	2970	4.3
	S1S2	31	1.3	31	1.6	2170	2.5	63	2.0	62	2.6	3600	4.5	53	2.3	53	2.8	2890	4.3
DHI	NO	52	5.6	51	6.1	3280	7.8	56	8.2	55	8.6	2500	9.5	54	11.3	54	11.5	2420	12.7
	C1	34	2.9	34	3.5	1950	3.7	41	4.6	40	5.3	1940	5.6	44	5.9	44	6.0	2140	7.4
	C1C2	34	2.3	34	2.7	2060	3.6	44	3.6	43	4.3	2110	4.6	46	5.0	46	5.8	2320	5.8
	C1C2C3	35	2.4	35	2.5	2160	4.3	44	4.5	43	5.0	2110	6.1	46	5.8	46	6.2	2320	7.3
	S1	29	1.4	29	1.7	1970	2.5	59	2.3	58	2.8	3290	4.7	53	2.4	53	2.9	2970	4.4
	S1S2	33	1.6	33	1.8	2290	2.9	63	1.9	62	2.5	3600	4.1	57	2.6	53	3.4	2890	4.6
EPG	NO	349	9.4	349	11.6	30780	25.7	283	8.7	283	11.3	23110	19.6	288	8.2	288	9.6	24040	21.6
	C1	97	4.2	97	5.3	7550	7.5	122	5.0	122	5.8	8930	10.3	145	5.9	145	7.4	11400	11.5
	C1C2	84	4.1	84	5.0	6510	7.4	118	4.8	118	6.1	8600	9.6	137	5.4	137	6.5	10620	11.3
	C1C2C3	85	3.9	85	4.4	6640	7.6	118	4.6	118	5.3	8600	9.2	137	5.4	137	6.8	10620	10.7
	S1	53	3.3	53	3.9	4020	5.4	93	3.9	93	5.0	6370	6.8	99	3.7	99	4.4	6970	7.2
	S1S2	54	3.4	54	3.7	4040	5.7	88	3.5	88	4.0	5870	6.5	93	4.2	93	5.0	6550	7.7

		Gen1						Gen2						Gen3					
		Separation		SEC Generation				Separation		SEC Generation				Separation		SEC Generation			
		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)	
Sep.	Shrinking	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time
EH	NO	102	538.5	102	539.8	8350	545.3	156	211.2	156	212.3	14470	224.9	116	291.9	116	293.6	10280	303.0
	C1	40	42.9	40	44.3	3070	45.0	32	47.6	32	47.8	2540	49.8	33	36.5	33	36.9	2410	39.0
	C1C2	44	30.6	44	30.8	3110	34.1	37	40.4	37	40.2	2580	43.2	38	30.7	38	31.8	2700	32.8
	C1C2C3	44	31.3	44	31.2	3110	34.8	36	41.0	36	41.2	2540	43.4	38	31.2	38	32.1	2700	33.5
	S1	56	11.4	56	11.9	3630	14.7	63	21.0	63	21.2	3620	24.1	53	12.7	53	13.7	3450	15.3
	S1S2	63	11.0	63	11.9	4220	14.6	71	20.4	71	20.4	4290	24.1	61	11.3	61	12.5	4180	14.5
DH	NO	80	30.3	80	31.9	4260	32.8	115	25.2	115	26.5	5380	29.3	73	18.7	72	19.8	3970	21.2
	C1	57	8.2	57	9.2	3030	10.4	89	7.1	89	8.8	4080	10.2	61	5.8	58	7.0	3030	8.2
	C1C2	60	6.5	60	7.5	3320	8.9	88	6.4	88	7.8	3980	10.3	62	5.4	59	6.2	3260	8.4
	C1C2C3	60	6.7	60	7.8	3320	8.9	87	6.0	87	7.3	3940	9.1	62	5.4	59	6.5	3260	8.1
	S1	69	3.4	67	4.7	4250	7.1	96	4.0	96	6.1	5090	8.0	74	2.7	73	3.8	4690	6.3
	S1S2	81	3.2	79	4.9	5250	7.1	108	3.4	108	5.7	6070	7.4	82	3.0	81	4.6	5360	7.4
DHI	NO	80	30.3	80	31.9	4260	32.8	115	25.2	115	26.5	5380	29.3	73	18.7	72	19.8	3970	21.2
	C1	57	10.0	57	11.0	3030	12.2	89	9.8	89	12.2	4080	12.2	61	7.7	58	8.7	3030	10.2
	C1C2	60	7.8	60	8.9	3320	10.2	88	8.3	88	9.7	3980	11.6	62	6.9	59	7.4	3260	10.1
	C1C2C3	60	8.3	60	9.2	3320	10.8	87	8.4	87	9.2	3940	11.9	62	7.3	59	8.8	3260	9.3
	S1	69	3.8	67	4.9	4250	7.6	96	4.0	96	5.4	5090	8.3	74	3.3	73	4.6	4690	7.3
	S1S2	81	3.7	79	5.3	5250	7.7	108	4.1	108	6.1	6070	8.9	82	3.0	81	4.8	5360	7.0
EPG	NO	690	24.0	690	32.5	62170	73.3	652	30.4	652	38.5	56770	81.7	493	23.5	493	28.7	44000	59.2
	C1	186	9.9	186	12.3	14770	21.9	222	14.1	222	16.6	16850	31.3	170	10.3	170	13.1	13660	20.7
	C1C2	167	9.4	167	11.6	13470	20.5	203	13.7	203	16.7	14900	28.5	154	9.1	154	11.8	12370	18.0
	C1C2C3	167	9.4	167	11.5	13470	20.7	202	13.8	202	16.5	14860	29.5	155	10.2	155	13.5	12490	19.0
	S1	117	7.6	117	9.9	9070	14.3	156	9.7	156	12.2	10890	20.3	113	6.1	113	7.7	8510	12.6
	S1S2	118	7.6	118	9.5	9270	14.8	153	9.2	153	11.6	10790	18.6	110	7.1	110	9.2	8240	13.2

		Gen1						Gen2						Gen3					
		Separation		SEC Generation				Separation		SEC Generation				Separation		SEC Generation			
		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)	
Sep.	Shrinking	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time
EH	NO	100	582.4	100	582.4	8930	589.9	27	441.1	27	442.9	1660	441.2	250	288.0	250	291.2	20440	307.4
	C1	28	83.7	28	83.6	2630	87.1	26	58.0	26	58.1	1710	59.9	62	50.2	62	51.3	4030	53.4
	C1C2	36	76.5	36	77.2	2880	79.3	36	43.2	36	43.1	1930	45.7	69	44.9	69	45.1	4260	49.4
	C1C2C3	36	77.2	36	78.5	2880	79.4	36	43.0	36	43.5	1930	45.0	69	45.9	69	46.8	4260	49.6
	S1	67	24.4	67	25.6	4650	27.6	72	22.0	72	22.0	4450	26.2	89	12.4	89	14.1	5130	15.6
	S1S2	70	23.0	70	24.4	4910	26.0	77	21.8	77	23.6	4970	24.5	111	11.2	111	13.0	6730	15.8
DH	NO	190	38.1	185	40.4	7330	44.2	173	32.1	170	34.0	7580	38.8	138	34.8	138	36.3	5450	39.3
	C1	131	11.3	127	13.3	6280	16.2	114	8.4	111	9.9	5530	13.4	111	9.1	111	10.7	4970	13.5
	C1C2	133	11.0	130	13.7	6480	15.5	113	7.5	110	9.5	5670	12.3	114	8.3	114	10.5	5290	12.4
	C1C2C3	133	10.1	130	12.6	6480	14.6	113	7.5	110	9.5	5660	12.1	114	8.0	114	10.3	5290	11.7
	S1	129	6.0	127	8.8	7570	11.5	120	4.4	118	7.0	7550	9.9	114	4.7	114	7.3	5870	9.3
	S1S2	130	5.5	128	8.2	7650	11.1	127	4.7	125	7.0	8110	11.4	139	3.9	139	6.8	7860	9.4
DHI	NO	190	38.1	185	40.4	7330	44.2	173	32.1	170	34.0	7580	38.8	138	34.8	138	36.3	5450	39.3
	C1	131	14.5	127	16.3	6280	19.1	114	11.4	111	13.2	5530	15.8	111	11.0	111	12.3	4970	15.1
	C1C2	133	13.7	130	16.0	6480	18.1	113	8.4	110	10.2	5670	12.6	114	9.3	114	10.9	5290	13.4
	C1C2C3	133	13.8	130	16.4	6480	17.9	113	9.5	110	12.0	5660	13.2	114	10.4	114	12.2	5290	14.5
	S1	129	6.8	127	9.4	7570	12.2	120	5.3	118	7.8	7550	11.2	114	5.3	114	7.9	5870	10.0
	S1S2	130	6.2	128	8.9	7650	11.1	132	5.5	125	7.2	8110	12.2	139	4.5	139	7.4	7860	9.9
EPG	NO	828	32.4	828	44.5	70130	84.7	803	29.9	803	41.2	68310	79.9	765	27.7	765	38.4	59420	70.0
	C1	280	15.1	280	19.9	21700	30.5	250	14.6	250	18.9	19280	28.2	280	13.5	280	17.7	18020	27.5
	C1C2	272	14.3	272	18.4	20960	29.1	233	13.2	233	16.8	17690	26.1	272	12.6	272	16.5	17370	25.4
	C1C2C3	272	14.1	272	18.2	20960	28.9	233	13.5	233	17.0	17680	26.5	272	13.3	272	17.4	17370	25.7
	S1	194	11.0	194	14.5	14790	20.8	175	9.1	175	12.7	13170	17.9	197	9.0	197	12.5	11890	17.1
	S1S2	189	10.3	189	13.1	14260	21.5	175	9.5	175	12.0	13270	19.4	184	8.5	184	11.4	12310	16.4

Table B.38: Number of obtained Q sets in separation, number of generated SECs when $k_{in} \times k_{out}$ is set to 1×1 and 10×10 and their running times by separation strategy, shrinking strategy and OP instance generation in vm1748.

		Gen1						Gen2						Gen3					
		Separation		SEC Generation				Separation		SEC Generation				Separation		SEC Generation			
		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)	
Sep.	Shrinking	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time
EH	NO	130	2198.8	130	2197.0	11200	2222.3	40	1068.1	40	1070.7	2970	1071.0	188	1140.2	188	1144.0	15540	1176.5
	C1	47	175.9	47	176.9	3250	181.9	28	227.5	28	228.1	1930	231.3	82	138.3	82	140.6	5740	147.2
	C1C2	63	152.6	63	155.1	3660	157.6	60	175.3	60	175.7	3000	181.2	86	114.5	86	116.0	5820	123.7
	C1C2C3	63	155.5	63	156.5	3660	161.9	60	175.1	60	176.4	3000	179.8	86	116.0	86	116.7	5820	126.0
	S1	112	95.9	112	98.9	6230	102.7	128	80.4	128	83.1	6030	87.0	82	32.5	82	34.1	4250	38.1
	S1S2	112	62.3	112	65.9	5980	67.2	143	70.2	143	73.6	7200	77.1	102	25.5	102	27.3	6070	32.6
DH	NO	186	78.2	185	82.5	7430	86.5	218	86.3	217	91.1	9940	95.5	143	63.4	143	66.3	6670	69.9
	C1	156	24.0	154	27.9	6770	30.6	185	50.5	184	54.8	8740	59.2	118	17.2	118	19.4	5300	23.0
	C1C2	164	20.6	162	24.2	7400	28.6	196	36.4	195	40.0	9580	47.0	118	14.1	118	17.3	5350	19.1
	C1C2C3	164	20.6	162	24.4	7400	28.7	197	37.2	196	40.9	9630	47.8	118	13.9	118	17.0	5350	19.0
	S1	175	12.6	173	16.8	8940	20.9	220	16.8	220	21.1	11170	28.1	113	9.1	112	11.9	5270	14.4
	S1S2	193	11.0	191	15.6	10340	20.1	238	15.4	238	21.0	12570	26.0	138	8.4	137	11.6	7440	15.1
DHI	NO	186	78.2	185	82.5	7430	86.5	218	86.3	217	91.1	9940	95.5	143	63.4	143	66.3	6670	69.9
	C1	156	32.5	154	35.6	6770	40.1	185	67.2	184	70.8	8740	76.2	118	21.6	118	24.5	5300	26.7
	C1C2	164	27.7	162	30.9	7400	36.2	196	50.8	195	55.8	9580	59.6	118	18.4	118	20.2	5350	25.2
	C1C2C3	164	29.4	162	32.7	7400	37.5	197	52.3	196	56.3	9630	62.2	118	20.0	118	22.2	5350	25.9
	S1	175	13.4	173	17.0	8940	21.9	220	18.6	220	24.3	11170	27.8	113	9.2	112	12.2	5270	14.0
	S1S2	193	12.6	191	16.6	10340	22.1	257	22.4	238	27.3	12570	34.4	144	10.5	137	13.8	7440	16.8
EPG	NO	1217	95.3	1217	120.5	109120	227.6	1140	64.2	1140	88.5	101750	161.5	1038	61.1	1038	81.1	92340	176.9
	C1	449	23.8	449	33.5	36440	64.3	513	26.8	513	38.0	41990	69.7	318	18.7	318	25.7	23820	45.5
	C1C2	426	22.4	426	32.2	33510	59.8	479	25.9	479	36.4	37390	63.3	304	19.0	304	25.8	22540	44.3
	C1C2C3	426	22.5	426	31.5	33510	60.6	478	24.8	478	36.2	37300	61.4	304	18.2	304	24.2	22540	45.2
	S1	312	21.5	312	29.1	23320	48.2	372	18.0	372	25.5	26540	43.9	207	13.3	207	17.6	14150	30.7
	S1S2	287	16.3	287	22.6	20970	40.0	355	18.2	355	26.8	24860	41.7	200	12.8	200	17.0	14280	27.8

Table B.39: Number of obtained Q sets in separation, number of generated SECs when $k_{in} \times k_{out}$ is set to 1×1 and 10×10 and their running times by separation strategy, shrinking strategy and OP instance generation in rl5934.

		Gen1						Gen2						Gen3					
		Separation		SEC Generation				Separation		SEC Generation				Separation		SEC Generation			
		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)	
Sep.	Shrinking	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time
EH	NO	21	7923.7	21	7914.3	1760	7942.1	60	12227.3	60	12243.7	5060	12233.7	307	7466.6	307	7486.2	28100	7530.7
	C1	19	991.6	19	997.7	1630	995.7	27	700.9	27	705.3	2410	707.9	57	877.0	57	879.1	3650	888.1
	C1C2	73	818.8	73	826.3	3510	829.4	71	622.5	71	629.4	4400	635.0	95	721.4	95	725.4	5300	737.8
	C1C2C3	73	816.8	73	825.1	3510	826.7	70	619.6	70	626.9	4370	631.1	94	722.7	94	729.4	5260	735.3
	S1	270	361.5	270	379.7	16660	393.6	285	528.9	285	548.7	17730	562.3	222	331.0	222	347.2	12170	354.4
	S1S2	315	334.5	315	356.5	20630	372.8	315	451.3	315	475.5	19890	485.0	270	275.3	270	293.9	16270	304.2
DH	NO	611	664.2	602	706.9	28970	728.0	665	575.3	660	623.0	30590	644.1	407	580.5	402	609.6	19770	621.7
	C1	415	126.3	403	153.6	22810	173.0	444	108.8	438	141.1	24660	158.3	291	72.5	287	92.5	15110	103.7
	C1C2	435	106.7	423	137.6	23640	155.2	460	97.6	454	131.2	25640	149.9	306	61.5	302	80.8	16060	94.3
	C1C2C3	435	106.3	423	137.5	23640	154.8	459	99.0	453	133.0	25650	149.7	304	61.6	300	81.1	15920	93.8
	S1	514	60.8	503	97.3	32830	117.6	519	60.8	514	97.9	33250	120.2	336	45.5	333	68.4	18080	79.2
	S1S2	549	58.1	538	95.5	35930	120.9	544	55.2	539	96.0	34370	115.7	378	40.4	375	66.8	21840	81.0
DHI	NO	611	664.2	602	706.9	28970	728.0	665	575.3	660	623.0	30590	644.1	407	580.5	402	609.6	19770	621.7
	C1	415	179.8	403	209.3	22810	227.2	444	188.8	438	220.5	24660	239.5	291	102.5	287	122.0	15110	132.6
	C1C2	435	149.1	423	179.8	23640	196.3	460	167.4	454	201.2	25640	218.1	306	83.6	302	105.4	16060	115.4
	C1C2C3	435	153.6	423	184.1	23640	203.5	459	173.0	453	205.3	25650	223.4	304	86.5	300	107.2	15920	119.2
	S1	514	67.7	503	104.4	32830	124.3	519	71.7	514	109.7	33250	130.4	336	49.7	333	71.7	18080	84.6
	S1S2	549	67.9	538	104.8	35930	129.3	544	66.5	539	105.9	34370	126.9	378	43.8	375	70.0	21840	82.8
EPG	NO	3661	734.5	3661	991.4	329310	1343.9	3434	777.9	3434	1044.9	303080	1370.5	3293	880.3	3293	1101.5	294840	1500.7
	C1	1011	103.6	1011	177.1	83650	282.0	1022	96.1	1022	172.9	83060	275.9	724	84.8	724	132.1	53500	202.8
	C1C2	927	91.8	927	159.9	75220	257.5	967	89.9	967	160.2	77540	253.4	673	78.3	673	123.5	48880	185.3
	C1C2C3	927	90.5	927	159.0	75220	258.2	964	87.3	964	159.4	77290	249.3	677	76.5	677	124.7	49080	184.3
	S1	753	62.7	753	116.9	58590	185.1	786	70.7	786	130.9	60470	205.3	536	54.2	536	90.3	36190	130.5
	S1S2	743	57.6	743	113.0	57470	176.9	746	68.1	746	123.7	56770	194.0	532	52.5	532	88.2	36470	127.7

Table B.40: Number of obtained Q sets in separation, number of generated SECs when $k_{in} \times k_{out}$ is set to 1×1 and 10×10 and their running times by separation strategy, shrinking strategy and OP instance generation in usa13509.

		Gen1						Gen2						Gen3					
		Separation		SEC Generation				Separation		SEC Generation				Separation		SEC Generation			
		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)	
Sep.	Shrinking	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time
EH	NO	1162	46201.1	1162	46550.1	108550	46915.2	1183	22163.0	1183	22528.4	109110	22903.0	676	33770.8	676	34109.3	60910	34008.7
	C1	95	4798.9	95	4827.7	7930	4857.0	229	1080.0	229	1138.9	19380	1221.3	180	2369.9	180	2410.5	15090	2467.2
	C1C2	94	4277.8	94	4312.3	7760	4320.3	223	984.5	223	1037.1	18770	1124.6	175	2008.6	175	2044.2	14590	2106.0
	C1C2C3	94	4277.8	94	4296.2	7760	4340.0	223	985.4	223	1038.4	18770	1120.6	175	2004.9	175	2047.0	14590	2097.2
	S1	286	1005.5	286	1075.0	19750	1117.2	286	394.0	286	459.9	20560	518.8	223	504.3	223	556.7	15600	593.8
	S1S2	431	782.6	431	872.9	32630	954.6	419	327.7	419	425.3	32410	501.3	314	384.8	314	451.3	23810	503.6
DH	NO	412	4608.6	412	4683.7	25490	4762.3	407	3107.9	407	3247.8	24230	3187.2	306	2828.2	306	2944.0	19650	2866.9
	C1	309	387.7	309	464.5	20750	483.1	271	256.6	271	319.7	17900	338.5	231	264.3	231	312.1	15750	331.9
	C1C2	307	351.5	307	427.7	20550	453.4	271	239.9	271	302.8	18050	329.1	229	235.1	229	284.0	15550	302.4
	C1C2C3	307	357.9	307	436.7	20550	459.0	271	247.3	271	308.6	18050	333.0	229	234.2	229	282.1	15550	305.0
	S1	291	179.8	291	247.8	19630	276.0	283	127.2	283	195.4	19720	216.2	219	126.5	219	178.1	14950	188.6
	S1S2	476	157.9	476	267.0	36010	316.4	446	110.8	446	215.6	34050	255.2	335	101.6	335	174.1	25060	200.8
DHI	NO	412	4608.6	412	4683.7	25490	4762.3	407	3107.9	407	3247.8	24230	3187.2	306	2828.2	306	2944.0	19650	2866.9
	C1	309	567.6	309	640.7	20750	664.7	271	396.0	271	457.5	17900	478.7	231	371.7	231	422.0	15750	439.3
	C1C2	307	507.7	307	587.5	20550	610.7	271	362.1	271	422.3	18050	447.6	229	315.2	229	365.9	15550	382.5
	C1C2C3	307	548.8	307	615.4	20550	652.4	271	398.7	271	466.3	18050	483.4	229	353.1	229	400.4	15550	421.8
	S1	291	192.3	291	260.1	19630	288.8	283	138.7	283	201.5	19720	228.8	219	135.5	219	180.5	14950	198.7
	S1S2	476	166.4	476	281.2	36010	319.9	446	122.3	446	218.9	34050	259.9	335	107.8	335	180.0	25060	206.1
EPG	NO	5367	4061.4	5367	5338.9	504100	6940.9	4430	2618.3	4430	3721.2	409520	4787.3	3283	1818.2	3283	2543.6	306330	3267.8
	C1	978	261.3	978	484.3	85220	742.0	840	178.0	840	374.2	71890	588.5	717	182.0	717	329.1	62610	530.0
	C1C2	949	242.5	949	474.3	82350	708.4	811	170.2	811	364.8	69650	579.1	678	167.9	678	318.9	58930	479.3
	C1C2C3	946	246.0	946	479.4	82020	711.5	814	170.7	814	358.4	69610	574.5	678	167.5	678	316.4	58930	480.8
	S1	588	164.6	588	307.3	48480	449.1	547	117.9	547	244.5	45350	368.7	432	110.1	432	198.6	35650	299.8
	S1S2	674	159.6	674	309.6	56260	482.3	628	114.4	628	260.8	52280	378.4	478	106.3	478	214.5	39740	303.7

Table B.41: Number of obtained Q sets in separation, number of generated SECs when $k_{in} \times k_{out}$ is set to 1×1 and 10×10 and their running times by separation strategy, shrinking strategy and OP instance generation in d15112.

		Gen1						Gen2						Gen3					
		Separation		SEC Generation				Separation		SEC Generation				Separation		SEC Generation			
		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)		(20 runs)		1x1 (10 runs)		10x10 (10 runs)	
Sep.	Shrinking	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time	#Q	Time	#SEC	Time	#SEC	Time
EH	NO	3284	42274.1	3284	43874.3	307840	43266.4	128	55626.9	128	55295.8	12080	56185.5	974	86877.2	974	87707.5	90920	86895.5
	C1	513	1864.8	513	2010.1	44790	2152.1	447	926.5	447	1048.0	38160	1138.6	127	4010.1	127	4026.4	11000	4095.6
	C1C2	495	1615.2	495	1763.4	43070	1879.5	435	837.7	435	959.8	37010	1026.9	131	3511.6	131	3542.8	11170	3580.4
	C1C2C3	495	1612.2	495	1758.9	43070	1878.9	435	839.7	435	963.4	37010	1037.3	131	3526.1	131	3541.4	11170	3609.6
	S1	348	466.2	348	564.1	28230	614.7	220	1349.8	220	1415.4	15440	1453.0	243	1035.1	243	1097.3	17380	1140.9
	S1S2	434	414.6	434	535.6	35780	591.0	320	1123.7	320	1219.4	24710	1257.3	342	874.3	342	979.0	26500	1015.7
DH	NO	354	4560.4	354	4691.7	23170	4658.4	401	3201.1	401	3341.2	24610	3315.4	361	5325.5	356	5591.6	24550	5295.0
	C1	242	362.2	242	428.9	17340	453.8	270	286.4	270	356.7	19180	383.3	269	487.5	264	569.3	18640	584.5
	C1C2	241	298.1	241	367.9	17290	385.3	266	254.9	266	331.4	18790	352.1	267	435.0	264	516.3	18660	530.1
	C1C2C3	241	300.5	241	373.6	17290	385.8	266	256.3	266	329.1	18790	355.5	267	437.0	264	514.2	18660	530.5
	S1	237	166.5	237	237.1	17340	249.4	278	132.0	278	202.2	20520	234.9	259	207.5	257	285.2	18230	303.9
	S1S2	380	144.1	380	253.8	30060	280.2	401	116.2	401	226.3	31170	263.4	389	182.7	387	301.0	29410	338.2
DHI	NO	354	4560.4	354	4691.7	23170	4658.4	401	3201.1	401	3341.2	24610	3315.4	361	5325.5	356	5591.6	24550	5295.0
	C1	242	519.5	242	581.7	17340	610.6	270	392.3	270	460.2	19180	491.0	269	710.6	264	783.9	18640	816.0
	C1C2	241	433.9	241	499.6	17290	521.6	266	348.1	266	417.6	18790	445.0	267	622.1	264	695.4	18660	724.1
	C1C2C3	241	460.3	241	526.5	17290	550.6	266	373.6	266	453.0	18790	472.9	267	675.2	264	760.0	18660	769.0
	S1	237	179.5	237	247.7	17340	263.5	278	144.6	278	220.5	20520	243.9	259	224.8	257	302.6	18230	322.9
	S1S2	380	150.6	380	250.2	30060	287.3	401	128.8	401	240.7	31170	271.0	389	199.1	387	315.5	29410	352.6
EPG	NO	5070	4634.5	5070	6114.2	479550	7891.6	4803	2714.3	4803	4082.5	447480	5683.1	4005	3486.2	4005	4604.9	375100	5701.0
	C1	809	257.9	809	483.6	72140	721.7	829	230.3	829	472.9	72890	716.8	869	297.0	869	539.5	75080	765.4
	C1C2	782	235.9	782	467.5	69660	662.9	785	213.9	785	428.8	68750	655.3	834	264.4	834	502.7	71410	699.9
	C1C2C3	780	248.8	780	475.8	69130	684.1	785	212.7	785	436.0	68750	650.3	834	268.4	834	500.2	71410	711.7
	S1	485	185.8	485	324.8	41720	458.0	545	155.8	545	309.6	46260	452.3	522	162.3	522	309.4	42500	408.4
	S1S2	555	181.3	555	334.7	47860	473.9	577	146.7	577	303.8	49440	450.9	561	159.9	561	311.2	47250	424.0

B.3 Chapter 4: Revisited Branch-and-Cut

B.3.1 Configuration of Components: Detailed Results

In this section, we show the detailed results of the alternative RB&C configurations by instances and generations. Each configuration has been executed five times with a 5-hour execution time limit. We show the obtained results of the configuration in terms of lower-bound values, LB, upper-bound values, UB, and time (in seconds) performance, Time. For the LB and UB, the obtained best value for each configuration (the maximum for LB and the minimum for the UB) is presented in the Best column. Regarding the Time, the Mean column shows the meantime of the five executions. The Gap column represents the relative distance to best-known value (highest Best value in the case of LB, and lowest Best in the case of UB and Mean in the case of Time, respectively).

Table B.42: pr76.

	Gen																	
	Gen1						Gen2						Gen3					
	LB		UB		Time		LB		UB		Time		LB		UB		Time	
Strategy	Best	Gap	Best	Gap	Mean	Gap	Best	Gap	Best	Gap	Mean	Gap	Best	Gap	Best	Gap	Mean	Gap
REFERENCE	49	0	49	0	0.04	123.66	2708	0	2708	0	1.13	90.94	2430	0	2430	0	1.03	39.55
- SRK	49	0	49	0	0.04	119.35	2708	0	2708	0	1.21	104.70	2430	0	2430	0	1.06	42.60
SRK=C1C2S3	49	0	49	0	0.04	111.83	2708	0	2708	0	1.38	133.98	2430	0	2430	0	0.90	21.84
- CC STRATS	49	0	49	0	0.04	124.73	2708	0	2708	0	1.13	90.57	2430	0	2430	0	1.04	39.74
- EPH BLOSSOM	49	0	49	0	0.03	64.52	2708	0	2708	0	1.44	143.58	2430	0	2430	0	0.74	0.00
- EGH BLOSSOM	49	0	49	0	0.02	0.00	2708	0	2708	0	1.22	106.29	2430	0	2430	0	0.97	30.56
+ FST BLOSSOM	49	0	49	0	0.09	398.92	2708	0	2708	0	1.32	123.29	2430	0	2430	0	0.85	14.47
- EDGE COVER	49	0	49	0	0.03	83.87	2708	0	2708	0	1.33	125.56	2430	0	2430	0	1.76	136.91
- CYCLE COVER	49	0	49	0	0.05	174.19	2708	0	2708	0	1.20	103.38	2430	0	2430	0	0.97	30.70
- PATH	49	0	49	0	0.04	116.13	2708	0	2708	0	1.39	135.40	2430	0	2430	0	0.79	7.02
+ VERTEX COVER	49	0	49	0	0.04	104.30	2708	0	2708	0	1.11	87.02	2430	0	2430	0	0.98	31.94
SEP: TWO SUBLOOPS	49	0	49	0	0.07	266.67	2708	0	2708	0	0.95	60.62	2430	0	2430	0	1.00	34.99
BRANCH HEUR=PB	49	0	49	0	0.05	175.27	2708	0	2708	0	0.59	0.00	2430	0	2430	0	1.41	90.47
BRANCH HEUR=VP - EA4OP	49	0	49	0	0.04	119.35	2708	0	2708	0	0.66	11.22	2430	0	2430	0	0.96	29.35

Table B.43: att532.

	Gen																	
	Gen1						Gen2						Gen3					
	LB		UB		Time		LB		UB		Time		LB		UB		Time	
Strategy	Best	Gap	Best	Gap	Mean	Gap	Best	Gap	Best	Gap	Mean	Gap	Best	Gap	Best	Gap	Mean	Gap
REFERENCE	363	0.00	363	0.00	359.51	1031.58	19633	0.06	19801	0.01	18000.00	0.00	15498	0.00	15498	0.00	166.80	29.99
- SRK	363	0.00	363	0.00	643.50	1925.50	19635	0.05	19800	0.01	18000.00	0.00	15498	0.00	15498	0.00	219.86	71.34
SRK=C1C2S3	363	0.00	363	0.00	120.89	280.53	19634	0.05	19800	0.01	18000.00	0.00	15498	0.00	15498	0.00	284.01	121.34
- CC STRATS	363	0.00	363	0.00	118.09	271.70	19633	0.06	19802	0.02	18000.00	0.00	15498	0.00	15498	0.00	2696.49	2001.40
- EPH BLOSSOM	363	0.00	363	0.00	31.77	0.00	19643	0.01	19801	0.01	18000.00	0.00	15498	0.00	15498	0.00	316.47	146.63
- EGH BLOSSOM	363	0.00	363	0.00	420.83	1224.61	19634	0.05	19801	0.01	18000.00	0.00	15498	0.00	15498	0.00	252.53	96.80
+ FST BLOSSOM	363	0.00	363	0.00	423.05	1231.61	19644	0.00	19801	0.01	18000.00	0.00	15498	0.00	15498	0.00	210.91	64.36
- EDGE COVER	363	0.00	363	0.00	176.79	456.47	19636	0.04	19800	0.01	18000.00	0.00	15498	0.00	15498	0.00	180.40	40.59
- CYCLE COVER	363	0.00	363	0.00	110.11	246.59	19642	0.01	19801	0.01	18000.00	0.00	15498	0.00	15498	0.00	221.88	72.91
- PATH	363	0.00	363	0.00	252.18	693.77	19629	0.08	19801	0.01	18000.00	0.00	15498	0.00	15498	0.00	212.04	65.25
+ VERTEX COVER	363	0.00	363	0.00	81.69	157.14	19637	0.04	19799	0.00	18000.00	0.00	15498	0.00	15498	0.00	305.62	138.18
SEP: TWO SUBLOOPS	363	0.00	363	0.00	300.17	844.81	19631	0.07	19801	0.01	18000.00	0.00	15498	0.00	15498	0.00	146.51	14.18
BRANCH HEUR=PB	363	0.00	363	0.00	190.93	500.97	19611	0.17	19800	0.01	18000.00	0.00	15498	0.00	15498	0.00	194.63	51.68
BRANCH HEUR=VP - EA4OP	363	0.00	363	0.00	270.75	752.20	19619	0.13	19801	0.01	18000.00	0.00	15498	0.00	15498	0.00	1000.74	679.89

Table B.44: vm1084.

	Gen																	
	Gen1						Gen2						Gen3					
	LB		UB		Time		LB		UB		Time		LB		UB		Time	
Strategy	Best	Gap	Best	Gap	Mean	Gap	Best	Gap	Best	Gap	Mean	Gap	Best	Gap	Best	Gap	Mean	Gap
REFERENCE	777	0.00	777	0.00	5378.5	144.79	40770	0.02	40954	0.02	18000.0	0.00	37669	0.00	37669	0.00	4735.9	150.57
- SRK	777	0.00	777	0.00	9969.9	353.75	40777	0.00	40952	0.01	18000.0	0.00	37669	0.00	37669	0.00	12469.7	559.74
SRK=C1C2S3	777	0.00	777	0.00	2731.9	24.34	40765	0.03	40953	0.02	18000.0	0.00	37669	0.00	37669	0.00	6725.9	255.85
- CC STRATS	777	0.00	777	0.00	4937.8	124.73	40772	0.01	40953	0.02	18000.0	0.00	37669	0.00	37669	0.00	5828.3	208.36
- EPH BLOSSOM	777	0.00	777	0.00	13669.6	522.14	40777	0.00	41006	0.15	18000.0	0.00	37665	0.01	37758	0.24	18000.0	852.33
- EGH BLOSSOM	777	0.00	777	0.00	7073.6	221.94	40773	0.01	40948	0.00	18000.0	0.00	37669	0.00	37669	0.00	8161.1	331.78
+ FST BLOSSOM	777	0.00	777	0.00	2197.2	0.00	40775	0.00	40946	0.00	18000.0	0.00	37669	0.00	37669	0.00	6688.2	253.86
- EDGE COVER	777	0.00	777	0.00	3303.3	50.34	40773	0.01	40954	0.02	18000.0	0.00	37669	0.00	37669	0.00	1890.1	0.00
- CYCLE COVER	777	0.00	777	0.00	4072.0	85.33	40775	0.00	40950	0.01	18000.0	0.00	37669	0.00	37669	0.00	4485.4	137.31
- PATH	777	0.00	777	0.00	4103.7	86.77	40775	0.00	40952	0.01	18000.0	0.00	37669	0.00	37669	0.00	7045.8	272.77
+ VERTEX COVER	777	0.00	777	0.00	3165.5	44.07	40777	0.00	40953	0.02	18000.0	0.00	37669	0.00	37669	0.00	8580.9	353.99
SEP: TWO SUBLOOPS	777	0.00	777	0.00	5145.1	134.17	40773	0.01	40950	0.01	18000.0	0.00	37669	0.00	37669	0.00	16501.5	773.05
BRANCH HEUR=PB	777	0.00	777	0.00	2596.3	18.16	40767	0.02	40955	0.02	18000.0	0.00	37669	0.00	37669	0.00	5133.0	171.57
BRANCH HEUR=VP - EA4OP	777	0.00	777	0.00	3767.8	71.48	40763	0.03	40956	0.02	18000.0	0.00	37669	0.00	37669	0.00	4421.5	133.93

Table B.45: rl1323.

	Gen																	
	Gen1						Gen2						Gen3					
	LB		UB		Time		LB		UB		Time		LB		UB		Time	
Strategy	Best	Gap	Best	Gap	Mean	Gap	Best	Gap	Best	Gap	Mean	Gap	Best	Gap	Best	Gap	Mean	Gap
REFERENCE	814	0.00	814	0.00	3565.7	10.26	43377	0.00	43454	0.18	18000.0	24.62	47162	0.11	47373	0.00	18000.0	0.00
- SRK	814	0.00	814	0.00	11747.3	263.25	43378	0.00	43452	0.17	18000.0	24.62	47195	0.04	47408	0.08	18000.0	0.00
SRK=C1C2S3	814	0.00	814	0.00	4039.8	24.92	43378	0.00	43457	0.18	18000.0	24.62	47212	0.01	47382	0.02	18000.0	0.00
- CC STRATS	814	0.00	814	0.00	5121.9	58.38	43378	0.00	43378	0.00	17145.6	18.71	47213	0.00	47386	0.03	18000.0	0.00
- EPH BLOSSOM	814	0.00	819	0.61	18000.0	456.60	43371	0.02	43543	0.38	18000.0	24.62	47075	0.30	47698	0.69	18000.0	0.00
- EGH BLOSSOM	814	0.00	814	0.00	4431.3	37.02	43377	0.00	43455	0.18	18000.0	24.62	47190	0.05	47394	0.05	18000.0	0.00
+ FST BLOSSOM	814	0.00	814	0.00	6341.4	96.09	43378	0.00	43378	0.00	15722.3	8.85	47200	0.03	47371	0.00	18000.0	0.00
- EDGE COVER	814	0.00	814	0.00	6401.6	97.95	43273	0.24	43456	0.18	18000.0	24.62	47109	0.22	47381	0.02	18000.0	0.00
- CYCLE COVER	814	0.00	814	0.00	7045.5	117.86	43378	0.00	43449	0.16	18000.0	24.62	47193	0.05	47385	0.03	18000.0	0.00
- PATH	814	0.00	814	0.00	3965.2	22.61	43378	0.00	43446	0.16	18000.0	24.62	47201	0.03	47379	0.02	18000.0	0.00
+ VERTEX COVER	814	0.00	814	0.00	3233.9	0.00	43377	0.00	43450	0.17	18000.0	24.62	47171	0.09	47379	0.02	18000.0	0.00
SEP: TWO SUBLOOPS	814	0.00	814	0.00	13939.5	331.04	43373	0.01	43451	0.17	18000.0	24.62	47196	0.04	47378	0.01	18000.0	0.00
BRANCH HEUR=PB	814	0.00	814	0.00	9743.9	201.30	43378	0.00	43378	0.00	16153.4	11.84	47215	0.00	47387	0.03	18000.0	0.00
BRANCH HEUR=VP - EA4OP	814	0.00	814	0.00	8707.5	169.25	43378	0.00	43449	0.16	18000.0	24.62	47195	0.04	47376	0.01	18000.0	0.00

Table B.46: vm1748.

	Gen																	
	Gen1						Gen2						Gen3					
	LB		UB		Time		LB		UB		Time		LB		UB		Time	
Strategy	Best	Gap	Best	Gap	Mean	Gap	Best	Gap	Best	Gap	Mean	Gap	Best	Gap	Best	Gap	Mean	Gap
REFERENCE	1276	0.23	1282	0.00	18000	0	68013	0.16	68305	0.01	18000	0	71903	0.01	72018	0.02	18000	0
- SRK	1271	0.63	1282	0.00	18000	0	67812	0.45	68306	0.01	18000	0	71853	0.08	72012	0.01	18000	0
SRK=C1C2S3	1278	0.08	1282	0.00	18000	0	67863	0.38	68306	0.01	18000	0	71887	0.03	72010	0.01	18000	0
- CC STRATS	1278	0.08	1282	0.00	18000	0	68016	0.15	68304	0.01	18000	0	71894	0.02	72012	0.01	18000	0
- EPH BLOSSOM	1273	0.47	1284	0.16	18000	0	67735	0.57	68460	0.23	18000	0	71755	0.21	72118	0.16	18000	0
- EGH BLOSSOM	1278	0.08	1282	0.00	18000	0	68029	0.14	68311	0.02	18000	0	71854	0.08	72016	0.02	18000	0
+ FST BLOSSOM	1279	0.00	1282	0.00	18000	0	67986	0.20	68300	0.00	18000	0	71773	0.19	72003	0.00	18000	0
- EDGE COVER	1272	0.55	1282	0.00	18000	0	67877	0.36	68306	0.01	18000	0	71873	0.05	72017	0.02	18000	0
- CYCLE COVER	1275	0.31	1282	0.00	18000	0	68055	0.10	68302	0.00	18000	0	71845	0.09	72014	0.02	18000	0
- PATH	1274	0.39	1282	0.00	18000	0	67831	0.43	68309	0.01	18000	0	71808	0.14	72013	0.01	18000	0
+ VERTEX COVER	1276	0.23	1282	0.00	18000	0	68032	0.13	68300	0.00	18000	0	71883	0.04	72016	0.02	18000	0
SEP: TWO SUBLOOPS	1276	0.23	1282	0.00	18000	0	67967	0.23	68314	0.02	18000	0	71830	0.11	72017	0.02	18000	0
BRANCH HEUR=PB	1274	0.39	1282	0.00	18000	0	67830	0.43	68300	0.00	18000	0	71779	0.18	72017	0.02	18000	0
BRANCH HEUR=VP - EA4OP	1278	0.08	1282	0.00	18000	0	67981	0.21	68307	0.01	18000	0	71890	0.03	72016	0.02	18000	0

B.3.2 Comparison with state-of-the-art Algorithms

In this appendix, we detail the experimental results for the four algorithms (FST B&C, EA4OP, ALNS and RB&C). Table B.47 shows the results for medium-sized instances of generation 1, Table B.48 for large-sized instances of generation 1, Table B.49 for medium-sized instances of generation 2, Table B.50 for large-sized instances of generation 2, Table B.51 for medium-sized instances of generation 3 and Table B.52 for large-sized instances of generation 3.

In the Best column, we show the global best-known lower and upper-bound values. For each algorithm, we detail the best LB, the goodness gap GGap, the best UB, and the meantime (in seconds). The GGap represents the relative distance between the algorithm's best LB and the global best-known LB. For the RB&C algorithm we also detail the optimality gap OGap which represents the relative distance between the obtained LB and UB by RB&C.

For each algorithm, generation and size, we have calculated the average gap and running time over the instances where a feasible solution was obtained by the algorithm. In those instances where the time limit was reached, a running time of 5 hours has been used. These averages are shown in the last row of the tables. The symbols in the tables mean the following:

$*$: best-known solution achieved

$-$: not comparable result

. : the code finished unexpectedly

Table B.47: Generation 1, $n \leq 400$

	Best		FST				EA4OP			ALNS			RB&C				
Instance	LB	UB	LB	GGap	UB	Time	LB	GGap	Time	LB	GGap	Time	LB	GGap	UB	OGap	Time
att48	31	31	**31**	*	**31**	**0.00**	**31**	*	0.25	**31**	*	6.77	**31**	*	**31**	*	**0.03**
gr48	31	31	**31**	*	**31**	**0.00**	**31**	*	0.13	**31**	*	9.99	**31**	*	**31**	*	**0.02**
hk48	30	30	**30**	*	**30**	**0.00**	**30**	*	0.24	**30**	*	7.20	**30**	*	**30**	*	**0.01**
eil51	29	29	**29**	*	**29**	**0.00**	**29**	*	0.24	**29**	*	9.51	**29**	*	**29**	*	**0.01**
berlin52	37	37	**37**	*	**37**	**0.00**	**37**	*	0.30	**37**	*	9.42	**37**	*	**37**	*	**0.02**
brazil58	46	46	**46**	*	**46**	**0.00**	**46**	*	1.00	**46**	*	9.13	**46**	*	**46**	*	0.07
st70	43	43	**43**	*	**43**	0.10	**43**	*	0.32	**43**	*	15.99	**43**	*	**43**	*	**0.05**
eil76	47	47	**47**	*	**47**	0.10	46	2.13	0.33	**47**	*	20.51	**47**	*	**47**	*	**0.04**
pr76	49	49	**49**	*	**49**	0.10	**49**	*	0.61	**49**	*	18.64	**49**	*	**49**	*	**0.06**
gr96	64	64	**64**	*	**64**	0.10	**64**	*	1.44	**64**	*	20.31	**64**	*	**64**	*	**0.08**
rat99	52	52	**52**	*	**52**	**0.40**	**52**	*	0.66	**52**	*	27.75	**52**	*	**52**	*	0.47
kroA100	56	56	**56**	*	**56**	**0.40**	55	1.79	0.34	**56**	*	34.75	**56**	*	**56**	*	**0.41**
kroB100	58	58	**58**	*	**58**	95.40	57	1.72	0.63	**58**	*	43.06	**58**	*	**58**	*	**0.27**
kroC100	56	56	**56**	*	**56**	0.40	**56**	*	0.48	**56**	*	34.32	**56**	*	**56**	*	**0.25**
kroD100	59	59	**59**	*	**59**	0.10	58	1.69	0.65	**59**	*	34.61	**59**	*	**59**	*	**0.09**
kroE100	57	57	**57**	*	**57**	159.20	**57**	*	0.50	**57**	*	32.26	**57**	*	**57**	*	**5.53**
rd100	61	61	**61**	*	**61**	0.20	**61**	*	0.74	**61**	*	29.49	**61**	*	**61**	*	**0.12**
eil101	64	64	**64**	*	**64**	0.10	**64**	*	0.79	**64**	*	31.73	**64**	*	**64**	*	**0.06**
lin105	66	66	**66**	*	**66**	**0.30**	**66**	*	1.42	**66**	*	32.11	**66**	*	**66**	*	0.48
pr107	54	54	**54**	*	**54**	0.30	**54**	*	0.93	**54**	*	78.46	**54**	*	**54**	*	**0.08**
gr120	75	75	**75**	*	**75**	**0.10**	74	1.33	1.20	**75**	*	29.58	**75**	*	**75**	*	0.28
pr124	75	75	**75**	*	**75**	**0.30**	**75**	*	1.11	**75**	*	49.64	**75**	*	**75**	*	0.35
bier127	103	103	**103**	*	**103**	**0.30**	**103**	*	1.18	**103**	*	40.84	**103**	*	**103**	*	0.38
pr136	71	71	**71**	*	**71**	**1.40**	**71**	*	0.96	**71**	*	29.97	**71**	*	**71**	*	1.75
gr137	81	81	**81**	*	**81**	1.50	78	3.70	3.44	**81**	*	59.21	**81**	*	**81**	*	**0.24**
pr144	77	77	**77**	*	**77**	**1.30**	**77**	*	2.61	**77**	*	87.82	**77**	*	**77**	*	1.46
kroA150	86	86	**86**	*	**86**	175.40	**86**	*	1.17	**86**	*	82.79	**86**	*	**86**	*	**33.87**
kroB150	87	87	**87**	*	**87**	**1.20**	86	1.15	1.00	**87**	*	61.64	**87**	*	**87**	*	2.21
pr152	77	77	**77**	*	**77**	1.40	**77**	*	3.64	**77**	*	91.38	**77**	*	**77**	*	**1.29**
u159	93	93	**93**	*	**93**	3.40	92	1.08	1.11	**93**	*	99.63	**93**	*	**93**	*	**1.82**
rat195	102	102	**102**	*	**102**	**2.60**	99	2.94	1.78	**102**	*	195.57	**102**	*	**102**	*	3.71
d198	123	123	**123**	*	**123**	**3.20**	**123**	*	6.68	**123**	*	65.57	**123**	*	**123**	*	5.28
kroA200	117	117	**117**	*	**117**	**1.20**	**117**	*	1.74	**117**	*	114.75	**117**	*	**117**	*	2.50
kroB200	119	119	**119**	*	**119**	14.10	**119**	*	1.67	**119**	*	86.58	**119**	*	**119**	*	**9.91**
gr202	145	145	**145**	*	**145**	12.70	**145**	*	6.89	**145**	*	187.56	**145**	*	**145**	*	**2.71**
ts225	124	124	**124**	*	**124**	**10216.30**	**124**	*	1.28	**124**	*	279.52	**124**	*	126	1.59	18000.00
tsp225	129	129	**129**	*	**129**	94.40	127	1.55	2.29	128	0.78	198.47	**129**	*	**129**	*	**4.31**
pr226	126	126	**126**	*	**126**	166.20	**126**	*	6.61	**126**	*	181.94	**126**	*	**126**	*	**107.69**
gr229	176	176	**176**	*	**176**	0.90	**176**	*	8.81	173	1.70	108.27	**176**	*	**176**	*	**0.32**
gil262	158	158	**158**	*	**158**	0.90	156	1.27	2.83	**158**	*	240.02	**158**	*	**158**	*	**0.35**
pr264	132	132	**132**	*	**132**	21.20	**132**	*	5.62	**132**	*	314.29	**132**	*	**132**	*	**3.92**
a280	147	147	**147**	*	**147**	**13.60**	143	2.72	3.00	144	2.04	239.06	**147**	*	**147**	*	40.65
pr299	162	162	**162**	*	**162**	111.50	160	1.23	3.12	**162**	*	410.90	**162**	*	**162**	*	**48.85**
lin318	205	205	**205**	*	**205**	22.40	202	1.46	7.15	203	0.98	294.23	**205**	*	**205**	*	**5.49**
rd400	239	239	**239**	*	**239**	37.40	234	2.09	6.59	237	0.84	422.56	**239**	*	**239**	*	**36.71**
average				*		**248.05**		0.62	2.12		0.14	99.51		*		0.04	407.20

Table B.48: Generation 1, $n > 400$

	Best		FST				EA4OP			ALNS			RB&C				
Instance	LB	UB	LB	GGap	UB	Time	LB	GGap	Time	LB	GGap	Time	LB	GGap	UB	OGap	Time
fl417	228	230	**228**	*	**230**	**18000.00**	224	1.75	11.84	**228**	*	1056.07	**228**	*	231	1.30	**18000.00**
gr431	350	350	**350**	*	**350**	139.90	349	0.29	32.84	347	0.86	533.55	**350**	*	**350**	*	**29.05**
pr439	313	313	**313**	*	**313**	833.30	310	0.96	9.92	307	1.92	1263.74	**313**	*	**313**	*	**414.00**
pcb442	251	251	**251**	*	**251**	14.90	244	2.79	6.94	249	0.80	1328.72	**251**	*	**251**	*	**7.21**
d493	320	320	**320**	*	**320**	347.30	315	1.56	19.10	317	0.94	1291.93	**320**	*	**320**	*	**13.37**
att532	363	363	**363**	*	**363**	593.00	347	4.41	23.14	359	1.10	1380.54	**363**	*	**363**	*	**312.50**
ali535	425	426	.	.	.	.	424	0.24	73.03	422	0.71	1846.10	**425**	*	**426**	0.23	**18000.00**
pa561	357	357	356	0.28	-	2103.60	348	2.52	23.18	346	3.08	1605.42	**357**	*	**357**	*	**245.42**
u574	354	354	**354**	*	**354**	61.40	344	2.82	17.93	347	1.98	1204.18	**354**	*	**354**	*	**24.00**
rat575	322	322	**322**	*	**322**	59.50	309	4.04	13.76	317	1.55	3109.65	**322**	*	**322**	*	**42.82**
p654	343	396	327	4.66	553	**18000.00**	336	2.04	28.89	**343**	*	10866.70	342	0.29	**396**	13.64	**18000.00**
d657	386	386	**386**	*	**386**	715.70	377	2.33	23.24	380	1.55	3152.17	**386**	*	**386**	*	**92.48**
gr666	503	503	**503**	*	**503**	634.20	497	1.19	109.54	486	3.38	660.30	**503**	*	**503**	*	**400.56**
u724	439	439	**439**	*	**439**	1077.10	429	2.28	27.77	434	1.14	4157.30	**439**	*	**439**	*	**188.61**
rat783	438	438	**438**	*	**438**	594.30	422	3.65	34.59	428	2.28	2962.52	**438**	*	**438**	*	**514.68**
dsj1000	656	656	.	.	.	.	632	3.66	81.20	630	3.96	17284.30	**656**	*	**656**	*	**3828.50**
pr1002	606	606	604	0.33	608	18000.00	572	5.61	45.92	581	4.13	18000.00	**606**	*	**606**	*	**4483.81**
u1060	660	660	.	.	.	.	627	5.00	90.04	644	2.42	18000.00	**660**	*	**660**	*	**16716.01**
vm1084	777	777	**777**	*	**777**	**4927.40**	770	0.90	56.29	765	1.54	18000.00	**777**	*	**777**	*	5012.60
pcb1173	675	675	.	.	.	.	633	6.22	60.65	652	3.41	18000.00	**675**	*	**675**	*	**6819.83**
d1291	715	715	.	.	.	.	646	9.65	434.87	699	2.24	18000.00	**715**	*	**715**	*	**7916.85**
rl1304	802	802	.	.	.	.	766	4.49	102.45	788	1.75	18000.00	**802**	*	**802**	*	**6269.39**
rl1323	814	814	811	0.37	846	18000.00	782	3.93	89.68	785	3.56	14585.10	**814**	*	**814**	*	**7740.17**
nrw1379	815	817	.	.	.	.	771	5.40	106.97	790	3.07	18000.00	**815**	*	**817**	0.24	**18000.00**
fl1400	1048	1084	909	13.26	1230	**18000.00**	1043	0.48	518.25	**1048**	*	18000.00	1003	4.29	**1084**	7.47	**18000.00**
u1432	754	764	.	.	.	.	738	2.12	121.46	749	0.66	14573.50	**754**	*	**764**	1.31	**18000.00**
fl1577	897	900	.	.	.	.	880	1.90	286.47	748	16.61	18000.00	**897**	*	**900**	0.33	**18000.00**
d1655	922	924	.	.	.	.	846	8.24	757.70	890	3.47	18000.00	**922**	*	**924**	0.22	**18000.00**
vm1748	1276	1282	873	31.58	.	**18000.00**	1246	2.35	178.50	1252	1.88	16959.80	**1276**	*	**1282**	0.47	**18000.00**
u1817	983	983	.	.	.	.	879	10.58	975.58	947	3.66	18000.00	**983**	*	**983**	*	**11226.88**
rl1889	1226	1226	890	27.41	1296	18000.00	1167	4.81	269.81	1156	5.71	18000.00	**1226**	*	**1226**	*	**17010.43**
d2103	1200	1200	.	.	.	.	1069	10.92	951.27	1171	2.42	18000.00	**1200**	*	**1200**	*	**15855.62**
u2152	1151	1151	.	.	.	.	1048	8.95	1350.23	1111	3.48	18000.00	**1151**	*	**1151**	*	**14703.25**
u2319	1170	1171	.	.	.	.	1167	0.26	423.26	**1170**	*	6088.42	**1170**	*	**1171**	0.09	**18000.00**
pr2392	1316	1415	1140	13.37	.	**18000.00**	1292	1.82	402.29	1294	1.67	18000.00	**1316**	*	**1415**	7.00	**18000.00**
pcb3038	1727	1730	.	.	.	.	1572	8.98	681.94	1626	5.85	18000.00	**1727**	*	**1730**	0.17	**18000.00**
fl3795	1965	2249	.	.	.	.	1815	7.63	2994.90	1818	7.48	18000.00	**1965**	*	**2249**	12.63	**18000.00**
fnl4461	2541	2570	.	.	.	.	2350	7.52	2462.65	2342	7.83	18000.00	**2541**	*	**2570**	1.13	**18000.00**
rl5915	3593	3786	.	.	.	.	3358	6.54	5361.54	3328	7.38	18000.00	**3593**	*	**3786**	5.10	**18000.00**
rl5934	3632	3752	.	.	.	.	3145	13.41	5382.25	3276	9.80	18000.00	**3632**	*	**3752**	3.20	**18000.00**
pla7397	5289	5657	.	.	.	.	5141	2.80	15981.78	5140	2.82	18000.00	**5289**	*	**5657**	6.51	**18000.00**
average				4.35		**7433.41**		4.32	990.82		3.12	11802.68		0.11		1.49	10387.02

Table B.49: Generation 2, $n \leq 400$

Instance	Best		FST				EA4OP			ALNS			RB&C				
	LB	UB	LB	GGap	UB	Time	LB	GGap	Time	LB	GGap	Time	LB	GGap	UB	OGap	Time
att48	1717	1717	**1717**	*	**1717**	**0.00**	**1717**	*	0.32	**1717**	*	6.77	**1717**	*	**1717**	*	**0.04**
gr48	1761	1761	**1761**	*	**1761**	**0.20**	1749	0.68	0.20	**1761**	*	7.87	**1761**	*	**1761**	*	1.32
hk48	1614	1614	**1614**	*	**1614**	**0.10**	**1614**	*	0.15	**1614**	*	7.19	**1614**	*	**1614**	*	**0.10**
eil51	1674	1674	**1674**	*	**1674**	**0.40**	1668	0.36	0.18	**1674**	*	10.13	**1674**	*	**1674**	*	0.96
berlin52	1897	1897	**1897**	*	**1897**	93.40	**1897**	*	0.35	**1897**	*	10.74	**1897**	*	**1897**	*	**3.23**
brazil58	2220	2220	**2220**	*	**2220**	**0.10**	2218	0.09	1.52	**2220**	*	12.32	**2220**	*	**2220**	*	0.46
st70	2286	2286	**2286**	*	**2286**	19.40	2285	0.04	0.31	**2286**	*	21.65	**2286**	*	**2286**	*	**1.77**
eil76	2550	2550	**2550**	*	**2550**	**0.10**	**2550**	*	0.43	**2550**	*	16.06	**2550**	*	**2550**	*	0.62
pr76	2708	2708	**2708**	*	**2708**	**0.40**	**2708**	*	0.48	**2708**	*	19.48	**2708**	*	**2708**	*	1.46
gr96	3396	3396	**3396**	*	**3396**	**1.70**	3394	0.06	1.44	3394	0.06	31.98	**3396**	*	**3396**	*	9.50
rat99	2944	2944	**2944**	*	**2944**	**0.90**	**2944**	*	0.49	**2944**	*	32.08	**2944**	*	**2944**	*	3.25
kroA100	3212	3212	**3212**	*	**3212**	0.90	**3212**	*	0.57	**3212**	*	32.85	**3212**	*	**3212**	*	**0.70**
kroB100	3241	3241	**3241**	*	**3241**	**6.70**	3238	0.09	0.52	3239	0.06	48.39	**3241**	*	**3241**	*	13.28
kroC100	2947	2947	**2947**	*	**2947**	85.60	2931	0.54	0.60	**2947**	*	39.27	**2947**	*	**2947**	*	**2.22**
kroD100	3307	3307	**3307**	*	**3307**	45.00	**3307**	*	0.65	**3307**	*	30.52	**3307**	*	**3307**	*	**3.62**
kroE100	3090	3090	**3090**	*	**3090**	230.10	3082	0.26	0.50	**3090**	*	39.57	**3090**	*	**3090**	*	**11.31**
rd100	3359	3359	**3359**	*	**3359**	**0.20**	**3359**	*	0.50	**3359**	*	30.80	**3359**	*	**3359**	*	0.36
eil101	3655	3655	**3655**	*	**3655**	153.00	**3655**	*	0.82	**3655**	*	26.19	**3655**	*	**3655**	*	**4.15**
lin105	3544	3544	**3544**	*	**3544**	67.30	3530	0.40	1.10	**3544**	*	36.22	**3544**	*	**3544**	*	**2.51**
pr107	2667	2667	**2667**	*	**2667**	0.60	**2667**	*	1.05	**2667**	*	69.67	**2667**	*	**2667**	*	**0.20**
gr120	4371	4371	**4371**	*	**4371**	35.80	4356	0.34	1.37	**4371**	*	40.41	**4371**	*	**4371**	*	**6.57**
pr124	3917	3917	**3917**	*	**3917**	**0.50**	3899	0.46	1.34	**3917**	*	55.25	**3917**	*	**3917**	*	1.07
bier127	5383	5383	**5383**	*	**5383**	58.80	5381	0.04	1.71	5366	0.32	23.01	**5383**	*	**5383**	*	**0.96**
pr136	4309	4309	**4309**	*	**4309**	2.10	**4309**	*	1.15	**4309**	*	35.63	**4309**	*	**4309**	*	**1.25**
gr137	4286	4286	**4286**	*	**4286**	196.90	4099	4.36	3.09	**4286**	*	639.80	**4286**	*	**4286**	*	**10.65**
pr144	4003	4003	**4003**	*	**4003**	90.40	3965	0.95	3.02	3969	0.85	100.20	**4003**	*	**4003**	*	**32.23**
kroA150	4918	4918	**4918**	*	**4918**	241.40	4902	0.33	1.26	**4918**	*	80.06	**4918**	*	**4918**	*	**60.43**
kroB150	4869	4869	**4869**	*	**4869**	24.80	**4869**	*	1.19	**4869**	*	61.96	**4869**	*	**4869**	*	**16.94**
pr152	4279	4279	**4279**	*	**4279**	2.20	4245	0.79	3.47	**4279**	*	67.41	**4279**	*	**4279**	*	**1.85**
u159	4960	4960	**4960**	*	**4960**	192.20	4941	0.38	1.44	4950	0.20	109.59	**4960**	*	**4960**	*	**14.96**
rat195	5791	5791	**5791**	*	**5791**	128.80	5703	1.52	1.55	5782	0.16	263.23	**5791**	*	**5791**	*	**46.09**
d198	6670	6670	**6670**	*	**6670**	**74.20**	6660	0.15	7.33	6661	0.13	88.47	**6670**	*	**6670**	*	298.24
kroA200	6547	6547	**6547**	*	**6547**	68.70	6534	0.20	1.71	**6547**	*	116.11	**6547**	*	**6547**	*	**16.18**
kroB200	6419	6419	**6419**	*	**6419**	34.70	6278	2.20	1.97	6413	0.09	189.98	**6419**	*	**6419**	*	**20.62**
gr202	7789	7789	**7789**	*	**7789**	**85.70**	**7789**	*	8.77	7719	0.90	188.27	**7789**	*	**7789**	*	139.90
ts225	6834	6834	**6834**	*	**6834**	**6.60**	6819	0.22	1.47	6782	0.76	394.00	**6834**	*	**6834**	*	95.22
tsp225	6987	6987	**6987**	*	**6987**	174.50	6936	0.73	1.87	6980	0.10	299.73	**6987**	*	**6987**	*	**54.09**
pr226	6662	6662	**6662**	*	**6662**	**74.10**	6658	0.06	7.29	**6662**	*	201.68	**6662**	*	**6662**	*	2894.81
gr229	9177	9177	**9177**	*	**9177**	182.60	9174	0.03	13.19	**9177**	*	1379.35	**9177**	*	**9177**	*	**16.67**
gil262	8321	8321	**8321**	*	**8321**	89.60	8175	1.75	3.47	8269	0.62	487.41	**8321**	*	**8321**	*	**64.63**
pr264	6654	6654	**6654**	*	**6654**	23.00	6173	7.23	5.94	**6654**	*	314.27	**6654**	*	**6654**	*	**13.33**
a280	8428	8428	**8428**	*	**8428**	**103.80**	8304	1.47	2.85	8404	0.28	215.31	**8428**	*	**8428**	*	519.95
pr299	9182	9182	**9182**	*	**9182**	**426.50**	9112	0.76	3.23	9147	0.38	393.12	**9182**	*	**9182**	*	623.34
lin318	10923	10923	**10923**	*	**10923**	862.40	10866	0.52	8.29	10801	1.12	370.64	**10923**	*	**10923**	*	**367.53**
rd400	13652	13652	**13652**	*	**13652**	**293.50**	13442	1.54	6.80	13562	0.66	1174.91	**13652**	*	**13652**	*	769.66
average				*		**92.89**		0.63	2.38		0.15	173.77		*		*	136.63

Table B.50: Generation 2, $n > 400$

Instance	Best LB	Best UB	FST LB	FST GGap	FST UB	FST Time	EA4OP LB	EA4OP GGap	EA4OP Time	ALNS LB	ALNS GGap	ALNS Time	RB&C LB	RB&C GGap	RB&C UB	RB&C OGap	RB&C Time
fl417	11933	12294	11894	0.33	**12294**	**18000.00**	11787	1.22	16.73	11923	0.08	2144.94	**11933**	*	12387	3.67	**18000.00**
gr431	18318	18318	**18318**	*	**18318**	**969.50**	18287	0.17	51.38	**18318**	*	2740.82	**18318**	*	**18318**	*	2809.41
pr439	16171	16171	**16171**	*	**16171**	**1298.30**	16085	0.53	11.77	16128	0.27	629.44	**16171**	*	**16171**	*	3765.86
pcb442	14484	14484	**14484**	*	**14484**	**6259.10**	14273	1.46	6.83	14411	0.50	4410.74	**14484**	*	**14484**	*	13760.94
d493	16995	17007	.	.	.	.	16729	1.57	17.15	16820	1.03	6231.42	**16995**	*	**17007**	0.07	**18000.00**
att532	19635	19800	19598	0.19	**19800**	**18000.00**	19265	1.88	23.43	19465	0.87	1564.89	**19635**	*	**19800**	0.83	**18000.00**
ali535	21954	21954	**21954**	*	**21954**	**2099.70**	21910	0.20	95.05	21761	0.88	1537.87	**21954**	*	21973	0.09	18000.00
pa561	19576	19576	**19576**	*	**19576**	**1487.10**	18894	3.48	23.45	19092	2.47	790.31	**19576**	*	**19576**	*	1961.95
u574	19351	19351	**19351**	*	**19351**	**612.50**	18966	1.99	16.33	19028	1.67	5389.10	**19351**	*	**19351**	*	1026.82
rat575	18251	18251	**18251**	*	**18251**	**931.10**	17705	2.99	14.97	17984	1.46	2089.02	**18251**	*	**18251**	*	9616.70
p654	17900	21566	17160	4.13	**21566**	**18000.00**	17821	0.44	42.82	**17900**	*	18000.00	17753	0.82	22248	20.20	**18000.00**
d657	21503	21503	**21503**	*	**21503**	2682.40	21162	1.59	22.90	21231	1.26	4161.44	**21503**	*	**21503**	*	**554.67**
gr666	26514	26569	.	.	.	.	26336	0.67	136.48	25971	2.05	1024.22	**26514**	*	**26569**	0.21	**18000.00**
u724	24223	24223	**24223**	*	**24223**	**5830.50**	23793	1.78	28.71	23878	1.42	5755.06	**24223**	*	**24223**	*	9829.42
rat783	25474	25474	.	.	.	.	24861	2.41	32.36	24987	1.91	6622.62	**25474**	*	**25474**	*	**12246.90**
dsj1000	35835	35915	35772	0.18	35917	**18000.00**	34463	3.83	83.34	34641	3.33	18000.00	**35835**	*	**35915**	0.22	**18000.00**
pr1002	33030	33092	27066	18.06	.	**18000.00**	31746	3.89	46.19	32120	2.76	18000.00	**33030**	*	**33092**	0.19	**18000.00**
u1060	36151	36291	.	.	.	.	35110	2.88	77.78	35284	2.40	18000.00	**36151**	*	**36291**	0.39	**18000.00**
vm1084	40777	40952	40687	0.22	40954	**18000.00**	40308	1.15	55.67	40240	1.32	18000.00	**40777**	*	**40952**	0.43	**18000.00**
pcb1173	37035	37100	.	.	.	.	35826	3.26	69.94	35946	2.94	18000.00	**37035**	*	**37100**	0.18	**18000.00**
d1291	37778	37854	.	.	.	.	35153	6.95	289.25	36815	2.55	18000.00	**37778**	*	**37854**	0.20	**18000.00**
rl1304	42275	42359	.	.	.	.	40561	4.05	97.68	40893	3.27	12853.40	**42275**	*	**42359**	0.20	**18000.00**
rl1323	43377	43450	43347	0.07	**43450**	**18000.00**	41459	4.42	89.78	41210	5.00	18000.00	**43377**	*	**43450**	0.17	**18000.00**
nrw1379	46676	46787	.	.	.	.	45602	2.30	117.51	45576	2.36	18000.00	**46676**	*	**46787**	0.24	**18000.00**
fl1400	56692	64298	53222	6.12	64726	**18000.00**	56258	0.77	794.15	**56692**	*	18000.00	54124	4.53	**64298**	15.82	**18000.00**
u1432	46946	47018	.	.	.	.	44810	4.55	100.91	44982	4.18	18000.00	**46946**	*	**47018**	0.15	**18000.00**
fl1577	45505	50154	.	.	.	.	**45505**	*	334.28	41148	9.57	18000.00	45326	0.39	**50154**	9.63	**18000.00**
d1655	49319	53083	.	.	.	.	47211	4.27	683.17	**49319**	*	18000.00	46158	6.41	**53083**	13.05	**18000.00**
vm1748	68042	68303	.	.	.	.	66685	1.99	195.85	66636	2.07	18000.00	**68042**	*	**68303**	0.38	**18000.00**
u1817	54245	54554	.	.	.	.	50366	7.15	734.39	51676	4.74	18000.00	**54245**	*	**54554**	0.57	**18000.00**
rl1889	63308	64425	52047	17.79	.	**18000.00**	60084	5.09	286.07	60928	3.76	18000.00	**63308**	*	**64425**	1.73	**18000.00**
d2103	63426	63426	.	.	.	.	57202	9.81	682.28	61636	2.82	18000.00	**63426**	*	**63426**	*	**16593.51**
u2152	64649	64775	53976	16.51	.	**18000.00**	60211	6.86	1164.38	61052	5.56	18000.00	**64649**	*	**64775**	0.19	**18000.00**
u2319	80914	81139	72790	10.04	.	**18000.00**	78102	3.48	447.06	77610	4.08	18000.00	**80914**	*	**81139**	0.28	**18000.00**
pr2392	72843	78237	64577	11.35	.	**18000.00**	71018	2.51	440.57	71851	1.36	18000.00	**72843**	*	**78237**	6.89	**18000.00**
pcb3038	97902	97995	83951	14.25	.	**18000.00**	91842	6.19	820.37	91457	6.58	18000.00	**97902**	*	**97995**	0.09	**18000.00**
fl3795	103397	142895	.	.	.	.	**103397**	*	4788.96	102642	0.73	18000.00	98998	4.25	**142895**	30.72	**18000.00**
fnl4461	147109	150189	.	.	.	.	140424	4.54	2618.15	135515	7.88	18000.00	**147109**	*	**150189**	2.05	**18000.00**
rl5915	184424	197729	.	.	.	.	176678	4.20	5512.40	173500	5.92	18000.00	**184424**	*	**197729**	6.73	**18000.00**
rl5934	187034	196805	.	.	.	.	171649	8.23	5757.80	166368	11.05	18000.00	**187034**	*	**196805**	4.96	**18000.00**
pla7397	281977	297246	.	.	.	.	272452	3.38	18000.00	266038	5.65	18000.00	**281977**	*	**297246**	5.14	**18000.00**
average				4.51		**11644.10**		3.13	1093.37		2.87	12827.93		0.40		3.06	15369.91

Table B.51: Generation 3, $n \leq 400$

	Best		FST				EA4OP			ALNS			RB&C				
Instance	LB	UB	LB	GGap	UB	Time	LB	GGap	Time	LB	GGap	Time	LB	GGap	UB	OGap	Time
att48	1049	1049	**1049**	*	**1049**	38.50	**1049**	*	0.259	**1049**	*	7.18	**1049**	*	**1049**	*	**1.17**
gr48	1480	1480	**1480**	*	**1480**	**0.20**	**1480**	*	0.13	**1480**	*	8.87	**1480**	*	**1480**	*	0.72
hk48	1764	1764	**1764**	*	**1764**	**0.00**	**1764**	*	0.215	**1764**	*	8.51	**1764**	*	**1764**	*	0.06
eil51	1399	1399	**1399**	*	**1399**	**0.20**	1398	0.07	0.222	**1399**	*	6.87	**1399**	*	**1399**	*	1.46
berlin52	1036	1036	**1036**	*	**1036**	124.70	1034	0.19	0.637	**1036**	*	12.84	**1036**	*	**1036**	*	**4.61**
brazil58	1702	1702	**1702**	*	**1702**	**0.00**	**1702**	*	0.711	**1702**	*	11.09	**1702**	*	**1702**	*	**0.02**
st70	2108	2108	**2108**	*	**2108**	**0.40**	**2108**	*	0.308	**2108**	*	9.65	**2108**	*	**2108**	*	0.49
eil76	2467	2467	**2467**	*	**2467**	**0.40**	**2467**	*	0.362	**2467**	*	20.48	**2467**	*	**2467**	*	2.96
pr76	2430	2430	**2430**	*	**2430**	**0.20**	**2430**	*	0.568	**2430**	*	20.43	**2430**	*	**2430**	*	1.07
gr96	3170	3170	**3170**	*	**3170**	61.50	3166	0.13	1.408	3166	0.13	15.22	**3170**	*	**3170**	*	**5.66**
rat99	2908	2908	**2908**	*	**2908**	4.90	-	-	-	-	-	-	**2908**	*	**2908**	*	**3.01**
kroA100	3211	3211	**3211**	*	**3211**	63.30	3180	0.97	0.379	**3211**	*	32.31	**3211**	*	**3211**	*	**1.81**
kroB100	2804	2804	**2804**	*	**2804**	0.60	2785	0.68	0.51	**2804**	*	35.83	**2804**	*	**2804**	*	**0.35**
kroC100	3155	3155	**3155**	*	**3155**	**1.50**	**3155**	*	0.439	**3155**	*	34.67	**3155**	*	**3155**	*	1.82
kroD100	3167	3167	**3167**	*	**3167**	10.70	3141	0.82	0.58	**3167**	*	31.08	**3167**	*	**3167**	*	**0.70**
kroE100	3049	3049	**3049**	*	**3049**	1.50	**3049**	*	0.471	**3049**	*	31.96	**3049**	*	**3049**	*	**1.36**
rd100	2926	2926	**2926**	*	**2926**	113.20	2923	0.10	0.482	**2926**	*	16.35	**2926**	*	**2926**	*	**23.20**
eil101	3345	3345	**3345**	*	**3345**	29.80	**3345**	*	0.564	**3345**	*	28.61	**3345**	*	**3345**	*	**1.37**
lin105	2986	2986	**2986**	*	**2986**	51.90	2973	0.44	2.094	**2986**	*	38.24	**2986**	*	**2986**	*	**16.02**
pr107	1877	1877	**1877**	*	**1877**	**660.90**	1802	4.00	0.816	**1877**	*	65.16	**1877**	*	**1877**	*	3297.37
gr120	3779	3779	**3779**	*	**3779**	**1.50**	3748	0.82	1.358	3777	0.05	37.94	**3779**	*	**3779**	*	2.65
pr124	3557	3557	**3557**	*	**3557**	**1021.50**	3455	2.87	0.882	**3557**	*	99.87	**3557**	*	**3557**	*	4507.38
bier127	2365	2365	**2365**	*	**2365**	79.90	2361	0.17	2.619	2361	0.17	49.9	**2365**	*	**2365**	*	**40.07**
pr136	4390	4390	**4390**	*	**4390**	86.70	**4390**	*	1.126	**4390**	*	61.84	**4390**	*	**4390**	*	**30.50**
gr137	3954	3954	**3954**	*	**3954**	**8.60**	**3954**	*	1.884	**3954**	*	637.09	**3954**	*	**3954**	*	14.01
pr144	3745	3745	**3745**	*	**3745**	**112.60**	3700	1.20	2.411	3744	0.03	112.92	**3745**	*	**3745**	*	116.68
kroA150	5039	5039	**5039**	*	**5039**	330.70	5019	0.40	1.07	5037	0.04	104.23	**5039**	*	**5039**	*	**46.43**
kroB150	5314	5314	**5314**	*	**5314**	107.60	**5314**	*	1.044	**5314**	*	63.05	**5314**	*	**5314**	*	**28.53**
pr152	3905	3905	**3905**	*	**3905**	1122.40	3902	0.08	3.625	3539	9.37	184.38	**3905**	*	**3905**	*	**83.51**
u159	5272	5272	**5272**	*	**5272**	52.20	**5272**	*	0.945	**5272**	*	94.27	**5272**	*	**5272**	*	**8.59**
rat195	6195	6195	**6195**	*	**6195**	49.90	-	-	-	-	-	-	**6195**	*	**6195**	*	**33.56**
d198	6320	6320	**6320**	*	**6320**	**286.10**	6290	0.47	7.145	**6320**	*	105.7	**6320**	*	**6320**	*	461.18
kroA200	6123	6123	**6123**	*	**6123**	122.30	6114	0.15	1.717	6118	0.08	232.2	**6123**	*	**6123**	*	**92.41**
kroB200	6266	6266	**6266**	*	**6266**	40.10	6213	0.85	1.775	**6266**	*	188.77	**6266**	*	**6266**	*	**3.87**
gr202	8616	8616	**8616**	*	**8616**	**224.80**	8605	0.13	10.452	8564	0.60	57.88	**8616**	*	**8616**	*	315.26
ts225	7575	7575	**7575**	*	**7575**	171.20	**7575**	*	1.136	**7575**	*	450.25	**7575**	*	**7575**	*	**6.62**
tsp225	7740	7740	**7740**	*	**7740**	150.30	-	-	-	-	-	-	**7740**	*	**7740**	*	**38.61**
pr226	6993	6993	**6993**	*	**6993**	**32.60**	6908	1.22	8.013	**6993**	*	177.59	**6993**	*	**6993**	*	1170.00
gr229	6328	6328	**6328**	*	**6328**	**10.20**	6297	0.49	11.655	**6328**	*	1298.8	**6328**	*	**6328**	*	42.63
gil262	9246	9246	**9246**	*	**9246**	133.40	9094	1.64	3.937	9210	0.39	649.54	**9246**	*	**9246**	*	**83.29**
pr264	8137	8137	**8137**	*	**8137**	**20.70**	8068	0.85	3.625	**8137**	*	357.8	**8137**	*	**8137**	*	186.59
a280	9774	9774	**9774**	*	**9774**	213.30	8684	11.15	3.22	8789	10.08	378.8	**9774**	*	**9774**	*	**126.80**
pr299	10343	10343	**10343**	*	**10343**	**363.60**	9959	3.71	3.952	10233	1.06	549.11	**10343**	*	**10343**	*	913.13
lin318	10368	10368	**10368**	*	**10368**	534.80	10273	0.92	6.327	10337	0.30	528.2	**10368**	*	**10368**	*	**327.58**
rd400	13223	13223	**13223**	*	**13223**	293.20	13088	1.02	7.738	13122	0.76	727.58	**13223**	*	**13223**	*	**214.40**
average				*		**149.66**		0.85	2.35		0.55	180.55		*		*	272.43

Table B.52: Generation 3, $n > 400$

	Best		FST				EA4OP			ALNS			RB&C				
Instance	LB	UB	LB	GGap	UB	Time	LB	GGap	Time	LB	GGap	Time	LB	GGap	UB	OGap	Time
fl417	14220	14220	**14220**	*	**14220**	**6227.60**	14186	0.24	12.449	**14220**	*	1131.05	14219	0.01	14387	1.17	18000.00
gr431	10911	10911	**10911**	*	**10911**	**1046.90**	10817	0.86	54.504	10907	0.04	2411.45	**10911**	*	**10911**	*	7814.17
pr439	15176	15296	15160	0.11	**15296**	**18000.00**	15097	0.52	10.96	15080	0.63	1328.74	**15176**	*	15331	1.01	**18000.00**
pcb442	14819	14819	**14819**	*	14839	18000.00	14522	2.00	6.578	14695	0.84	1192.19	**14819**	*	**14819**	*	**11574.76**
d493	25167	25188	**25167**	*	**25188**	**18000.00**	24981	0.74	19.182	24849	1.26	3829.32	**25167**	*	25195	0.11	**18000.00**
att532	15498	15498	**15498**	*	**15498**	933.20	15342	1.01	22.747	15335	1.05	4533.36	**15498**	*	**15498**	*	**318.44**
ali535	9414	9472	.	.	.	.	9328	0.91	94.089	9308	1.13	13313.5	**9414**	*	**9472**	0.61	**18000.00**
pa561	14482	14482	**14482**	*	**14482**	10543.80	-	-	-	-	-	-	**14482**	*	**14482**	*	**2539.41**
u574	20064	20064	**20064**	*	**20064**	**1409.30**	19691	1.86	19.766	19841	1.11	1671.01	**20064**	*	**20064**	*	2693.59
rat575	20109	20109	**20109**	*	**20109**	1426.50	-	-	-	-	-	-	**20109**	*	**20109**	*	**929.99**
p654	24492	24518	**24492**	*	31914	**18000.00**	24130	1.48	18.541	24427	0.27	7543.02	**24492**	*	**24518**	0.11	**18000.00**
d657	24562	24562	**24562**	*	**24562**	**4053.30**	23772	3.22	21.887	23829	2.98	4600.87	**24562**	*	**24562**	*	8777.39
gr666	17023	17048	17020	0.02	**17048**	**18000.00**	16902	0.71	143.868	16709	1.84	2734.75	**17023**	*	17060	0.22	**18000.00**
u724	28348	28348	**28348**	*	**28348**	**5870.60**	27932	1.47	29.263	28033	1.11	12058.6	**28348**	*	**28348**	*	10332.54
rat783	27566	27566	**27566**	*	**27566**	7232.30	-	-	-	-	-	-	**27566**	*	**27566**	*	**3812.98**
dsj1000	31434	31454	.	.	.	.	30943	1.56	79.179	31040	1.25	15962	**31434**	*	**31454**	0.06	**18000.00**
pr1002	39526	39526	39449	0.19	39545	18000.00	38762	1.93	47.303	38502	2.59	18000	**39526**	*	**39526**	*	**13955.69**
u1060	37492	37569	.	.	.	.	36570	2.46	75.876	36598	2.38	18000	**37492**	*	**37569**	0.20	**18000.00**
vm1084	37669	37669	37653	0.04	37694	18000.00	37508	0.43	54.207	37178	1.30	3286.89	**37669**	*	**37669**	*	**8710.50**
pcb1173	41257	41257	.	.	.	.	40069	2.88	66.158	40513	1.80	18000	**41257**	*	**41257**	*	**15133.74**
d1291	41509	42153	30106	27.47	.	**18000.00**	38132	8.14	299.865	39919	3.83	18000	**41509**	*	**42153**	1.53	**18000.00**
rl1304	41881	42075	40478	3.35	.	**18000.00**	41214	1.59	81.109	41679	0.48	18000	**41881**	*	**42075**	0.46	**18000.00**
rl1323	47213	47384	44458	5.84	.	**18000.00**	46641	1.21	93.526	45500	3.63	8544.44	**47213**	*	**47384**	0.36	**18000.00**
nrw1379	42920	42975	.	.	.	.	-	-	-	-	-	-	**42920**	*	**42975**	0.13	**18000.00**
fl1400	57470	59491	54792	4.66	67053	**18000.00**	57226	0.42	599.811	**57470**	*	18000	54661	4.89	**59491**	8.12	**18000.00**
u1432	47778	47895	.	.	.	.	46657	2.35	138.016	47242	1.12	18000	**47778**	*	**47895**	0.24	**18000.00**
fl1577	45935	48809	.	.	.	.	45692	0.53	295.615	**45935**	*	18000	45768	0.36	**48809**	6.23	**18000.00**
d1655	62048	62945	51168	17.53	.	**18000.00**	58728	5.35	674.247	60956	1.76	18000	**62048**	*	**62945**	1.43	**18000.00**
vm1748	71885	72010	68979	4.04	.	**18000.00**	70958	1.29	225.29	71244	0.89	18000	**71885**	*	**72010**	0.17	**18000.00**
u1817	63639	67670	52186	18.00	.	**18000.00**	**63639**	*	1302.347	63016	0.98	18000	63618	0.03	**67670**	5.99	**18000.00**
rl1889	70065	71106	43374	38.09	.	**18000.00**	68422	2.34	244.973	68096	2.81	18000	**70065**	*	**71106**	1.46	**18000.00**
d2103	82787	82973	76035	8.16	.	**18000.00**	77333	6.59	1168.899	81081	2.06	18000	**82787**	*	**82973**	0.22	**18000.00**
u2152	74007	78066	52091	29.61	.	**18000.00**	73400	0.82	1619.609	72733	1.72	18000	**74007**	*	**78066**	5.20	**18000.00**
u2319	79351	81050	**79351**	*	81619	**18000.00**	78113	1.56	569.758	79130	0.28	18000	79343	0.01	**81050**	2.11	**18000.00**
pr2392	85409	90261	60225	29.49	.	**18000.00**	84094	1.54	422.734	85084	0.38	18000	**85409**	*	**90261**	5.38	**18000.00**
pcb3038	106928	112006	96356	9.89	.	**18000.00**	104667	2.11	917.386	105337	1.49	18000	**106928**	*	**112006**	4.53	**18000.00**
fl3795	97707	116792	.	.	.	.	**97707**	*	3158.887	95580	2.18	18000	89218	8.69	**116792**	23.61	**18000.00**
fnl4461	146995	152562	.	.	.	.	-	-	-	-	-	-	**146995**	*	**152562**	3.65	**18000.00**
rl5915	203695	217366	.	.	.	.	199336	2.14	5593.23	201814	0.92	18000	**203695**	*	**217366**	6.29	**18000.00**
rl5934	212021	229405	.	.	.	.	207385	2.19	5881.87	203667	3.94	18000	**212021**	*	**229405**	7.58	**18000.00**
pla7397	322285	334885	.	.	.	.	320744	0.48	18000	312645	2.99	18000	**322285**	*	**334885**	3.76	**18000.00**
average				6.78		**13749.78**		1.80	1168.44		1.47	12837.26		0.34		2.24	14843.74