UNIVERSITÉ DE MONTRÉAL

OPTIMISATION DE L'INTÉGRATION DES REQUÊTES DE RÉSEAUX VIRTUELS DANS UN ENVIRONNEMENT MULTICLOUD

MARIEME DIALLO
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INFORMATIQUE)
DÉCEMBRE 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

OPTIMISATION DE L'INTÉGRATION DES REQUÊTES DE RÉSEAUX VIRTUELS DANS UN ENVIRONNEMENT MULTICLOUD

présentée par : DIALLO Marieme
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

Mme BELLAÏCHE Martine, Ph. D., présidente
M. PIERRE Samuel, Ph. D., membre et directeur de recherche
M. QUINTERO Alejandro, Doctorat, membre et codirecteur de recherche
M. BEAUBRUN Ronald, Ph. D., membre
M. AJIB Wessam, Ph. D., membre externe

DÉDICACE

À mes grand-parents, Amadou, Mayram, Jean et Henriette,
partis si tôt...

À mes parents, Alassane et Maïmouna,
pour leur amour inconditionnel et leur foi en moi...

À mon mari, mon partenaire de vie, mon ami, Driss,
pour son amour et son soutien incommensurables...

À mes soeurs, Khadijetou et Oumou, et mon frère, Cherif,
mes bradios de coeur pour toute la vie...

À ma niece, Amina, et mon neveu, Mouhamed,
mes petits bradios d'amour...

Je vous aime de tout mon coeur...

REMERCIEMENTS

Tout d'abord, j'aimerais adresser ma profonde et sincère gratitude à mes directeurs de recherche, M. Samuel Pierre et M. Alejandro Quintero, pour la qualité de leur encadrement, leur disponibilité et leur support académique et financier. Je les remercie grandement pour leurs conseils avisés et leur niveau d'exigence qui m'ont permis de faire valoir davantage le meilleur de moi-même tout au long de ce travail de recherche. Je porte une reconnaissance particulière à M. Pierre pour m'avoir donné l'opportunité et la grande confiance de réaliser ce parcours doctoral sous sa supervision. L'estime et l'admiration que j'ai pour M. Pierre n'ont fait que s'agrandir durant toutes ces années, tant j'ai été marquée par le modèle de réussite qu'il incarne, sa profonde humanité, son humilité et l'attention parternelle qu'il m'a portée. Je tiens également à souligner le réel plaisir que j'ai eu à travailler avec M. Quintero, que je ne remercierai jamais assez pour ses encouragements, son immense sympathie et sa bonne humeur constante peu importe les difficultés, en plus de son professionalisme et de son soutien technique qui ont été si précieux pour l'avancement de mon travail.

J'aimerais pareillement exprimer toute l'affection que j'ai pour Mme Martine Bellaïche, que je n'oublierai jamais pour avoir tant égayé mes journées difficiles, par ses chaleureux conseils, son soutien moral et amical.

Je remercie aussi M. Ronald Beaubrun et M. Wessam Ajib pour l'intérêt porté à mon travail en acceptant de participer à ce jury.

Je témoigne également ma gratitude à l'Agence Canadienne de Développement International (ACDI) pour m'avoir offert la possibilité de poursuivre des études universitaires au Canada dans le cadre du Programme Canadien de Bourse de la Francophonie (PCBF), ainsi qu'à la fondation Roasters pour m'avoir soutenue en tant qu'étudiante internationale, et à la fondation *Education for Development and Democracy Initiative* (EDDI) avec qui tout a commencé.

Je manifeste, par ailleurs, ma reconnaissance à tout le personnel administratif et technique du département de génie informatique et génie logiciel. Je porte une attention particulière à tous mes collègues et amis du LAboratoire de recherche en Réseautique et Informatique Mobile (LARIM). Il n'y a pas assez de mots pour exprimer toute ma gratitude pour m'avoir offert une deuxième maison si conviviale, où la solidarité, l'entraide et l'amitié m'ont permis de surmonter les difficultés de tous les jours durant toutes ces années. Une mention toute particulière à Georges Abou-Khalil, Valérie Justafort et Rana Farah, pour leur amitié, leur disponibilité et les longues heures de discussion dans une ambiance des plus plaisantes, et à Lamia El Garoui pour sa grande bonté envers moi.

Je ne puis oublier, dans cette litanie de reconnaissance et de remerciements, tous les étudiants de l'École Polytechnique à qui j'ai eu le vrai bonheur d'enseigner durant toutes ces années, et qui m'ont offert le plaisir de remporter à de nombreuses fois le prix de meilleure chargée de cours dans mon domaine de génie.

Un grand merci à toute ma grande famille Aïdara-Diallo, ma belle famille et mes ami(e)s pour leur présence dans ma vie. Un merci du fond du cœur à mon époux, Driss Badiane, qui m'a sans cesse accompagnée dans cette aventure laborieuse avec tant de patience, d'amour et de soutien moral.

Enfin, je suis inestimablement redevable envers mon père, Alassane Robert Diallo, et ma mère, Maïmouna Aïdara. Je ne les remercierai jamais assez pour les valeurs telles que la fierté, l'honneur, la simplicité, l'abnégation, le courage, la dignité et la foi qu'ils ont su placer en moi. Sans leur amour, leurs innombrables sacrifices et leurs prières, je ne serais pas aussi forte et épanouie que je ne le suis aujourd'hui. Merci de m'avoir donné le courage d'affronter la vie et de toujours m'élever au-delà de moi-même. Merci d'avoir été mes sources d'inspiration et mes modèles de perfection sur tous les aspects de la vie. Que Dieu les benisse car il n'y a rien sur cette terre qui puisse leur rendre ce qu'ils méritent...

RÉSUMÉ

De nos jours, l'Infrastructure-service ou Infrastructure as a Service (IaaS) est devenue le modèle de service du Cloud Computing le plus largement adopté. Dans ce modèle d'affaires, un fournisseur de service ou Service Provider (SP) peut louer, à partir d'un ou de plusieurs fournisseurs d'infrastructure ou Cloud Providers (CPs), des ressources physiques proposées en tant que services (calcul, stockage, accès réseau, routage, etc.). Ces derniers sont encapsulés dans des machines virtuelles ou Virtual Machines (VMs), interconnectées et assemblées sous forme de requête de réseau virtuel ou Virual Network Request (VNR), dans le but de créer des réseaux virtuels hétérogènes offrant des applications et des services personnalisés à des utilisateurs finaux.

Malgré son adoption largement réussie, le modèle IaaS reste toujours confronté à un défi fondamental en matière de gestion de ressources, qui consiste en l'optimisation de l'intégration efficace et dynamique des VNRs dans les infrastructures sous-jacentes distribuées et partagées. En effet, des ressources hétérogènes doivent être efficacement allouées afin de pouvoir héberger les VMs dans des centres de données ou data centers (DCs) spécifiques, et de faire router les liaisons virtuelles ou Virtual Links (VLs), représentant le trafic échangé entre les VMs interconnectées, sur des chemins appropriés entre les DCs. Cette allocation de ressources et de services vise généralement à satisfaire des contraintes de performance, de Qualité de Service (QdS), de sécurité et de localisation géographique, imposées par le SP. Dans le contexte de la virtualisation de réseau, ce problème est connu NP-difficile, sous le nom d'intégration de réseau virtuel ou Virtual Network Embedding (VNE), qui n'a été abordé que récemment dans la littérature dans le cadre d'un réseau multiCloud, où les infrastructures Cloud sous-jacents appartiennent à différents CPs indépendants.

Le VNE dans un environnement multiCloud ajoute plus de complexité et des défis d'évolutivitité au problème, car l'ensemble du processus nécessite une approche de résolution hiérarchique, dans laquelle deux phases principales d'opération sont réalisées, chacune ayant des objectifs différents selon les acteurs : la phase de partitionnement des VNRs à travers le réseau multiCloud, suivie de la phase d'intégration des segments de VNRs dans les infrastructures intraCloud sélectionnées. Dans la première phase réalisée indirectement par le SP, ce dernier mandate généralement un fournisseur de réseau virtuel ou Virtual Network Provider (VNP). Le VNP agit en tant que service de courtage virtuel pour le compte du SP, afin de sélectionner adéquatement des CPs capables de répondre efficacement aux objectifs et exigences du SP, puis partitionne les VNRs en plusieurs segments. Dans la deuxième phase,

qui correspond notamment au problème bien connu du VNE dans le cadre d'un seul CP et qui a été largement abordé dans des travaux de recherche antérieurs, chaque CP sélectionné utilise une approche d'hébergement adéquate pour intégrer les segments de VNRs qui lui sont attribués dans son réseau intraCloud.

Cette thèse traite le problème émergent du VNE dans un environnement multiCloud, dont les principales contributions sont apportées à la phase de partitionnement des VNRs, puisqu'une approche d'hébergement existante est généralement adoptée pour résoudre la phase d'intégration intraCloud des segments de VNRs. La phase de partitionnement des VNRs, également classée comme un problème NP-difficile, fait face à de nombreux défis liés à la non-interopérabilité entre les CPs, à leurs politiques d'accès restrictives aux informations internes à leur réseau, aux caractéristiques dynamiques de l'environnement multiCloud et à la nature complexe du problème. Il devient alors difficile pour le VNP de sélectionner de manière optimale les services Cloud appropriés et de générer des solutions d'intégration de VNRs répondant au mieux aux objectifs du SP. D'autre part, les quelques approches existantes proposant des solutions au problème, outre le fait de supposer que les CPs divulgueraient certaines informations privées, ont principalement visé à uniquement minimiser les coûts d'approvisionnement de ressources pour le SP durant la phase de partitionnement, ou soit se sont uniquement intéressées à des solutions basées sur les performances des VNRs, sans aucune considération économique. Ces approches de résolution ne donnent donc pas au SP la possibilité de sélectionner des services Cloud en fonction du niveau de performance et de QdS désiré, tout en réduisant au minimum les dépenses induites.

Afin de relever les défis mentionnés ci-dessus, nous proposons dans cette thèse un cadre hiérarchique d'optimisation de l'intégration des VNRs dans un réseau multiCloud offrant des services sous le modèle IaaS. Le cadre proposé inclut notamment l'élaboration d'une approche stratégique de partitionnement des VNRs et l'adoption d'une approche subséquente d'hébergement intraCloud des segments de VNRs résultants. Le modèle de VNE multiCloud mis en œuvre prend conjointement en compte les exigences des VMs et du trafic inter-VMs, ainsi que la nature dynamique de l'environnement multiCloud et l'aspect complexe du problème. L'objectif est de partitionner efficacement les VNRs et d'optimiser la performance et la QdS des segments de VNRs résultants qui seront intégrés dans les infrastructures Cloud sélectionnées, tout en minimisant les coûts monétaires d'approvisionnement de ressources pour le SP. Le travail global de la thèse a été réalisé en trois principaux volets, chacun faisant l'objet d'un article scientifique.

Dans le premier volet du travail, nous proposons un cadre de VNE dans un environnement multiCloud, qui vise à maximiser le taux d'acceptation des VNRs et à optimiser les perfor-

mances et la QdS de ces dernières selon des exigences du SP, tout en réduisant également le délai de communication inter-VMs. À cette fin, nous concevons d'abord une stratégie de partitionnement des VNRs qui prend en compte les ressources serveur et réseau nécessaires à la performance respective des VMs et des VLs. De plus, notre approche stratégique considère plusieurs aspects liés à la QdS, spécifiés par le SP dans les caractéristiques des VMs à héberger, ainsi que l'empreinte géographique souhaitée des VMs et le délai maximal de trafic toléré entre des VMs communicantes. Par ailleurs, afin de fournir une meilleure efficacité de la stratégie de partitionnement, nous avons mené une recherche approfondie sur les informations auxquelles le VNP peut accéder dans le but d'enrichir sa visibilité sur l'environnement multiCloud, tout en respectant la confidentialité des CPs. Sur la base des informations collectées et des spécifications du SP, nous avons défini une stratégie d'approvisionnement de ressources à l'aide des techniques de programmation linéaire en nombres entiers, afin de formaliser le problème sous forme d'un modèle mathématique avec des contraintes. Par la suite, pour résoudre la phase d'intégration intraCloud des segments de VNRs, une approche multiobjectif d'hébergement de ressources est adoptée, dans le but de minimiser les coûts en ressources et en énergie pour chaque CP sélectionné, ainsi que le délai de communication dans la réseau intraCloud. Dans cette première étape de notre travail, chacune des phases de partitionnement et d'hébergement est résolue de manière optimale avec la méthode exacte, en utilisant AMPL et l'outil d'optimisation CPLEX. Le modèle proposé est évalué à l'aide de simulations numériques et est comparé à d'autres approches partitionnement de VNRs de la littérature utilisant la même solution pour la phase d'hébergement intraCloud. Les résultats obtenus démontrent l'efficacité de la stratégie de partitionnement proposée selon plusieurs critères de performance, incluant le taux d'acceptation des VNRs et le délai de communication qui sont notamment améliorés respectivement d'environ 15.1 % et 18.5 %, tout en évitant les violations de la QdS qui sont également réduites de 27.8 %.

Le VNE dans un environnement multiCloud est un problème complexe à résoudre en raison de la nature NP-difficile des deux phases de partitionnement et d'hébergement intraCloud. Par conséquent, les méthodes exactes proposées dans le premier volet du travail ne sont pas idéales pour résoudre un tel problème, vu qu'elles ne peuvent évoluer raisonnablement qu'avec des instances de petite taille du problème, en raison de l'augmentation exponentielle du temps de résolution. De ce fait, il devient nécessaire de recourir à des méthodes d'approximation rapides, basées sur les (méta) heuristiques, afin d'obtenir des solutions (presque) optimales en un temps de résolution polynomial pour de grandes instances du problème. À cette fin, nous proposons dans le deuxième volet de notre travail une approche de résolution de la phase partitionnement des VNRs basée sur la Recherche Taboue ou Tabu Search (TS). La méthode TS est choisie pour son efficacité notoire à résoudre divers problèmes d'optimisation traitant

des aspects d'évolutivité. L'adaptation de l'algorithme de TS proposée exécute un processus de Recherche Locale ou Local Search (LS) et un mécanisme de mémoire à long terme (Diversification). De plus, au lieu d'évaluer entièrement chaque configuration de solution obtenue itérativement, ce qui est très couteux en temps de calcul, nous définissons des fonctions de calcul de gains exprimant la différence entre le coût d'une solution actuelle et celui d'une solution voisine. Cette procédure est plus compliquée à implémenter, mais elle nous permet de réduire considérablement l'ordre de complexité du temps de calcul de $O(n^2)$ à $O(n)$. L'approche heuristique proposée est implémentée en C++ et est évaluée par des simulations numériques, avec différents scénarios de test examinés et en considérant des tailles de VNRs pouvant s'étendre jusqu'à 500 VMs. Des phases préliminaires de paramétrage sont d'abord effectuées, afin de définir les bonnes valeurs des paramètres de l'algorithme de TS en fonction de la taille des instances du problème. Ensuite, la performance de l'algorithme est évaluée en termes de qualité des solutions obtenues et de temps de résolution. Pour les VNRs de petite taille, les résultats de comparaison avec la méthode exacte montrent que notre approche approximative, en un temps de calcul linéairement réduit, est capable de générer presque à tous les coups la solution optimale, avec une distance de coût à moins de 2.97 % en moyenne. Pour les VNRs de grande taille, l'algorithme est tout aussi efficace, générant des coûts de solution très proches du coût maximum jamais atteint par l'heuristique, avec un écart en moyenne de 0.17 %. Enfin, nous résolvons l'approche d'hébergement intraCloud adoptée en utilisant AMPL/CPLEX et un algorithme de descente itérée, respectivement pour les VNRs de petite et de grande taille. Les résultats prouvent que notre approche heuristique est aussi efficace que la méthode exacte pour prendre les meilleures décisions possibles de partitionnement de VNRs, en fonction des mêmes mesures de performance que celles considérées dans le premier volet de notre travail.

Dans le dernier volet de notre travail, le cadre de VNE multiCloud proposé est étendu afin de prendre en compte, outre la performance et la QdS des VNRs, la minimisation des coûts monétaires d'approvisionnement de ressources pour le SP. Pour la phase de partitionnement, le nouveau modèle est d'abord résolu avec la méthode exacte sur des petites instances du problème. Ensuite, pour des instances de taille plus importante, une approche de résolution hybride basée sur une combinaison intégrative des métaheuristiques de colonies de fourmis ou Ant Colony Optimization (ACO) et de TS est proposée. Une telle hybridation nous permet de tirer profit des capacités d'adaptation efficaces d'ACO dans le domaine de l'optimisation, et de l'efficacité de TS pour éviter les cycles et les pièges d'optima locaux. L'algorithme de TS, incluant un mécanisme de Diversification, est utilisé comme opérateur de LS améliorateur, exécuté sur la meilleure solution construite par les fourmis à chaque itération. De plus, afin d'accélérer considérablement l'évaluation d'une nouvelle configuration de solution construite

à chaque étape par une fourmi, des fonctions de calcul de variations sont aussi définies dans ce volet pour uniquement évaluer la différence de coût générée par l'ajout d'un nouveau composant à la configuration partielle d'une solution actuelle. Les résultats de simulations montrent que le processus de LS hybridé avec ACO améliore significativement la qualité des solutions obtenues par rapport à l'algorithme d'ACO simple, mais au prix d'un temps de résolution relativement plus élevé. L'algorithme de TS proposé dans le deuxième volet du travail est également utilisé pour résoudre le nouveau modèle de partitionnement, mais donne des solutions légèrement de moins bonne qualité en moyenne comparé à l'approche hybride. Avec des VNRs de petite taille, les deux algorithmes hybride et de TS génèrent des solutions très proches de la solution optimale, avec une distance moyenne à moins de 3.42 % et 4.18 %, respectivement. Des résultats similaires sont observés pour les VNRs de plus grande taille, les solutions étant en moyenne meilleures avec l'approche hybride qu'avec l'algorithme de TS (l'algorithme d'ACO simple donnant lui dans certains cas des solutions irréalisables). Pour la phase d'intégration intraCloud, le modèle est également étendu afin de maximiser les profits de chaque CP sélectionné. D'autres expériences sont effectuées sur des VNRs de petite taille, dans le but d'évaluer et de comparer le taux d'acceptation des VNRs, le coût d'approvisionnement pour le SP et le profit du CP, dans les cas où seule l'optimisation des performances des VNRs est prise en compte dans la stratégie de partitionnement (comme c'est le cas pour les deux premiers volet du travail), ou seule la réduction des coûts d'approvisionnement pour le SP est ciblée (dans la plupart des cas, dû au fait qu'aucune information sur les ressources n'est divulguée au VNP, à l'exception de leur prix unitaire). Les résultats montrent que le taux d'acceptation est plus bas si uniquement la réduction des coûts est l'objectif principal du SP. Par ailleurs, le nouveau cadre de VNE étendu améliore le rapport performance/coût d'environ 8 % pour le SP, et le profit d'environ 67.2 % pour les CPs.

Le cadre de VNE multiCloud proposé dans cette thèse permet d'offrir au SP un modèle varié de sélection des ressources et services Cloud les mieux adaptés en fonction de ses intérêts et de ses objectifs. Aussi, les approches de résolution approximatives proposées offrent un très bon compromis entre la qualité des solutions obtenues et le temps de résolution. Quant aux CPs, les résultats obtenus pourraient les inciter à divulguer davantage d'informations sur les ressources au VNP et à être plus ouverts en matière d'interopérabilité, de manière à optimiser l'utilisation de leurs ressources, et de ce fait d'augmenter le taux d'acceptation et les profits.

ABSTRACT

Nowadays, the Infrastructure as a Service (IaaS) has become the most widely adopted cloud service model. In this business paradigm, a Service Provider (SP) can lease, from one or more Cloud Providers (CPs), infrastructure layer resources (processing, storage, network access, routing services, etc.) packaged into interconnected virtual machines (VMs) and assembled as a virtual network request (VNR), in order to build heterogeneous virtual networks that will offer customized services and applications to its end users.

Despite its successful adoption, the IaaS model faces a fundamental resource management challenge lying in the efficient and dynamic embedding of VNRs onto distributed and shared substrate infrastructures. Heterogenous resources need to be efficiently allocated to host VMs in specific substrate data centers (DCs) and to route virtual links (VLs), representing the exchanged traffic between interconnected VMs, onto suitable substrate paths between the hosting DCs, in order to satisfy performance, Quality of Service (QoS), security and geographical location constraints imposed by the SP. In the context of network virtualization, this issue is usually referred to as the NP-hard Virtual Network Embedding (VNE) problem, which has been only recently addressed in the literature within a multicloud network, where the substrate infrastructures are owned by different and independent CPs.

Such a context adds more complexity and scalability issues, since the whole VNE process requires a hierarchical resolution approach, where two major phases of operation are performed, each of them having different purposes according to the acting player: the multicloud VNRs splitting phase, followed by the intra-cloud VNR segments mapping phase. In the first phase played indirectly by the SP, the latter generally mandates a Virtual Network Provider (VNP), which acts as a virtual brokerage service on behalf of the SP, in order to select eligible CPs based on the SP's goals and requirements, and split the VNRs into segments. In the second phase, which corresponds to the well known VNE within a single CP largely addressed in past research works, each selected CP uses a mapping approach to embed the assigned VNR segments into its intra-cloud network.

This thesis addresses the emergent problem of VNE across a multicloud environment, where the main contributions are focused on the splitting phase, since generally an existing intra-cloud embedding approach is adopted to solve the mapping phase. The VNRs splitting phase, usually reduced to the NP-hard multiway separator problem, deals with many challenges related to the non-interoperability between CPs, their restrictive access policies to internal network information, the dynamic character of the multicloud environment and the

complex nature of the problem. This makes difficult for the VNP to select optimally the right cloud services and to generate embedding solutions which must best satisfy the SP's purposes. On the other hand, the few existing approaches for the multicloud VNRs splitting problem, besides assuming that CPs would share some private information, mostly only aim at minimizing the resource provisioning cost for the SP during the splitting phase, or only propose performance-based splitting approaches without any economic aspects considered. This may not give opportunities to the SP to select the cloud services based on the desired performance and QoS, while at the same time minimizing the resulting expenditures.

In order to address the challenges mentioned above, we propose in this thesis a hierarchical optimization framework for embedding VNRs across a multicloud IaaS-based network, which includes a VNRs splitting strategy and an intra-cloud VNR segments mapping approach. The proposed framework considers jointly the VMs and inter-VMs traffic requirements in the embedding process, as well as the dynamic nature of the multiCloud environment and the scalability issues of the problem. The objective is to efficiently split the VNRs and to optimize the performance and QoS of the resulting VNR segments that are mapped onto the selected cloud infrastructures, while minimizing the resource provisioning expenditures for the SP. The work is conducted in three major stages, each of them leading to a scientific article.

First, we propose a multicloud VNE framework which aims to maximize the acceptance rate and to optimize the best possible the performance and QoS of VNRs based on the SP's requirements, while minimizing the overall inter-VMs communication delay. To this end, we design a VNRs splitting strategy to solve the splitting phase, which considers all computational and networking resources that can be required for the performance of VMs and VLs. In addition, our splitting strategy takes into account several QoS aspects specified by the SP, such as the desired features and geographical footprint of VMs, as well as the maximum traffic delay allowed on communicating VMs. Moreover, in order to provide a better efficiency of the proposed splitting strategy, we conducted a thorough investigation of the information that the VNP can access, to further enhance its visibility on the multicloud environment while complying with the CPs' privacy. Based on the collected information and the SP's specifications, we define a resource provisioning strategy using an Integer Linear Program (ILP) to formalize the VNRs splitting phase as a mathematical model with constraints. To solve the VNR segments mapping phase, we adopt a multi-objective resource mapping approach which aims to minimize the resources and environmental costs of embedding for each selected CP, as well as the intra-cloud communication delay. Both splitting and mapping phases in this first stage of our work are solved optimally with the exact method using AMPL with CPLEX as optimizing solver. The proposed splitting strategy is compared through

simulations to other baselines approaches performing the same intra-cloud resource mapping solution. The results obtained demonstrate the efficiency of our strategy according to several performance criteria, including the VNRs acceptance rate and the overall delay which are in particular respectively improved by about 15.1 % and 18.5 %, while avoiding QoS violations which are also reduced by about 27.8 %.

The multicloud VNE is a complex problem to solve due to the NP-hard nature of both the splitting and mapping phases. Therefore, exact methods are not suitable to solve such a problem since they can only scale up to small sized instances due to the exponential increase of the resolution time. Overcoming this issue needs the development of fast approximate resolution methods based on (meta) heuristics, in order to obtain (near) optimal solutions in polynomial time for large instances of the problem. To this end, we propose in the second stage of our work an optimization approach based on TS for solving the VNRs splitting problem. TS is chosen for its notorious effectiveness for various optimization problems that deal with scalability issues. The proposed TS-based splitting approach executes a LS process and an enhancing long-term memory mechanism (Diversification). Furthermore, instead of entirely evaluating each solution configuration obtained iteratively, which is very expensive in calculation time, we define gain functions expressing the cost difference generated by moving from a current solution to a neighbor solution. This procedure is more complicated to perform, but it allows us to decrease significantly the computing time complexity order from $O(n^2)$ to $O(n)$. The proposed approach is implemented in C++ and evaluated through simulations with various test scenarios examined and considering VNRs with up to 500 VMs. First, preliminary parameterization tests are conducted in order to define the good parameter values of the TS algorithm depending on the size of the instances' problem. Then, the performance of the algorithm is evaluated in terms of quality of solution obtained and resolution time. For small sized VNRs, comparison results with the exact method show that our approximate approach is able to quasi-always generate the optimal solution in a linearly reduced computing time, with a maximum cost distance on average of about 2.97 %. For large sized VNRs, the algorithm is just as effective, generating solution costs very close to the maximal cost ever found by the heuristic, with an average cost distance of about 0.17 %. Finally, we solve the adopted mapping approach using AMPL/CPLEX and an iterated descent algorithm, respectively for small and large sized VNRs. Results proves that our heuristic approach is as effective as the exact method to perform the best possible splitting decisions based on the same performance metrics considered in the first stage of our work.

In the last stage of our work, the proposed multicloud VNE framework is extended in order to consider, in addition to the performance and QoS of VNRs, the minimization of the resource provisioning expenditures for the SP. For the splitting phase, the extended ILP

model is first solved with the exact method with small instances of the problem. Then, for larger instances, we propose a hybrid approach based on an integrative combination of Ant Colony Optimization (ACO) and TS metaheuristics. Such a hybridization allows us to take advantage on ACO's effective adaptive capabilities in optimization domain and on TS's efficiency in avoiding cycles and local optima traps. TS algorithm, along with the Diversification mechanism implemented, is used as an enhancing LS operator performed on the best solution constructed by the ants at each iteration. Also, in order to significantly speed up the evaluation of a new solution configuration built by an ant, variation functions are defined to only evaluate the cost difference generated by adding a new solution component to a current partial solution configuration. Simulation results show that the LS process hybridized with ACO significantly improves the quality of solutions compared to the simple ACO, but at the price of more resolution time needed. The TS algorithm proposed in the second stage of the work is also used to solve the extended splitting model and gives slightly less good solutions on average compared to the hybrid approach. With small sized VNRs, both approaches generate solutions very close to the optimal solution, with an average maximum distance of about 3.42 % and 4.18 %, respectively for the hybrid approach and the TS algorithm. Same results are noticed for the large sized VNRs, with solutions on average better with the hybrid approach than with the TS algorithm (the simple ACO giving in some cases unfeasible solutions). For the mapping phase, the model is extended for maximizing each selected CP's profit. Other experiments are performed on small sized VNRs in order to evaluate and compare the VNRs acceptance rate, the SP's expenditure and the CP's profit, in the cases where only the VNRs performance optimization is considered in the splitting strategy (as it is with the two first stages of the work), or only the cost minimization is targeted (in most cases because any resource information is disclosed to the VNP except their unit price). Results show that the acceptance rate is lower with only the cost minimization as SP's purpose. On the other hand, the new extended VNE framework improves the ratio performance/expenditure by about 8 % for the SP and the profit by about 67.2 % for the CPs.

The multicloud VNE framework proposed in this thesis allows to offer to the SP a varied model of selection of the right cloud products and services depending on its interests and goals. Also, the proposed approximate resolution approaches offer a very good tradeoff between quality of solutions obtained and resolution time. As for the CPs, the results obtained could encourage them to disclose more resource information to the VNP and to be more open to interoperability, so they can improve the resource utilization efficiency and therefore increase the acceptance rate and the profits.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

ACDI	Agence Canadienne de Développement International
ACO	Ant Colony Optimization
AMPL	A Mathematical Programming Language
AWS	Amazon Web Service
CP	Cloud Provider
CPs	Cloud Providers
CPU	Central Processing Unit
DC	Data Center
DCs	Data Centers
EC2	Elastic Cloud Compute
EDDI	Education for Development and Democracy Initiative
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
IaaS	Infrastructure as a Service
ILP	Integer Linear Programming
ILS	Iterated Local Search
LARIM	LAboratoire de recherche en Réseautique et Informatique Mobile
LS	Local Search
MILP	Mixed-Integer Linear Programming
MPC	Multi-Party Computation
NIST	National Institute of Standards and Technology
NSFNet	National Science Foundation Network
NP-difficile	Non déterministe Polynomial-difficile
PaaS	Platform as a Service
PCBF	Programme Canadien de Bourse de la Francophonie
PoP	Point of Presence
PoPs	Points of Presence
PSO	Particle Swarm Optimization
PUE	Power Usage Effectiveness
QdS	Qualité de Service
QoS	Quality of Service
SA	Simulated Annealing
SaaS	Software as a Service

SLA	Service Level Agreement
SP	Service Provider
TIC	Technologies de l'Information et de la Communication
TL	Time Limit
TS	Tabu Search
VL	Virtual Link
VLs	Virtual Links
VLAN	Virtual Local Area Network
VM	Virtual Machine
VMs	Virtual Machines
VNE	Virtual Network Embedding
VNP	Virtual Network Provider
VNR	Virtual Network Request
VNRs	Virtual Network Requests
VNS	Variable Neighborhood Search

CHAPITRE 1 INTRODUCTION

Au cours des dernières années, le Cloud Computing s'est rapidement imposé sur le marché des Technologies de l'Information et de la Communication (TIC) pour devenir l'une des plateformes les plus importantes de prestation à distance de services informatiques et réseautiques (Murugesan et Bojanova, 2016; Armbrust *et al.*, 2010). Avec un taux de croissance annuel estimé de 13.4% prévu d'ici 2020 (Statista, 2018), cette solution technologique utilitaire et rentable est largement adoptée par les petites entreprises, qui y voient un modèle d'économie à grande échelle. Ces dernières peuvent désormais externaliser la gestion de leurs applications à des prestataires de services qui mettent à leur disposition, à travers Internet, une large gamme de ressources configurables facturées sur utilisation, sans qu'ils aient à se soucier de la gestion complexe et onéreuse de l'infrastructure sous-jacente.

Parmi les modèles de services introduits par le Cloud Computing (Mell et Grance, 2009; Murugesan et Bojanova, 2016), l'Infrastructure-service ou Infrastructure as a Service (IaaS) est celui qui connaît le taux de croissance annuel le plus rapide sur le marché, estimé à environ 23.3% entre 2016 à 2020 (Statista, 2018). Dans ce modèle d'affaires, le rôle de prestataire de services est détenu notamment par le fournisseur d'infrastructure ou Cloud Provider (CP), qui est chargé de gérer et de déployer l'infrastructure et l'équipement réseau virtualisés du Cloud, et le fournisseur de service ou Service Provider (SP), qui loue, sous forme de requêtes de réseau virtuel ou Virtual Network Requests (VNRs), des ressources informatiques et réseautiques à partir d'un ou de plusieurs CPs, afin d'offrir des applications et des services personnalisés à des utilisateurs finaux.

Malgré son implantation largement réussie sur le marché des TIC, le modèle IaaS fait face à un défi fondamental quant à la gestion et à l'allocation automatique des ressources, dans un environnement réseau virtualisé dynamique, évolutif et à forte convergence de divers types d'applications à héberger et de locataires de services davantage exigeants à satisfaire (Manvi et Shyam, 2014; Zhang *et al.*, 2016a). Un tel enjeu nécessite la mise en œuvre de mécanismes efficaces d'hébergement des ressources demandées dans les réseaux d'infrastructures sous-jacents, afin d'optimiser, à moindres coûts pour le SP et aux meilleurs profits pour le CP, la performance et la qualité des services offerts, tout en assurant leur disponibilité, leur continuité et leur fiabilité.

Dans le contexte de la virtualisation de réseau, ce processus est généralement associé au problème complexe d'intégration de réseau virtuel ou Virtual Network Embedding (VNE) (Fischer *et al.*, 2013; Zhang *et al.*, 2016a), qui n'était jusqu'à récemment appliqué que dans

le cadre d'un seul CP. Toutefois, le désir du SP à déployer ses applications et services à une clientèle de plus en plus vaste et géographiquement distribuée, nécessite l'intégration des VNRs dans un environnement multiCloud (Rafael *et al.*, 2012; Heilig *et al.*, 2016), constitué d'infrastructures hétérogènes appartenant à plusieurs CPs indépendants.

Dans cette thèse, nous traitons le problème émergent du VNE dans un environnement multiCloud, en proposant un cadre hiérarchique d'optimisation de l'intégration des VNRs à travers plusieurs infrastructures Cloud. Ce cadre inclut une première phase stratégique de partitionnement des requêtes parmi les multiples CPs, suivant les objectifs et les exigences du SP, et une phase subséquente d'hébergement des segments de requêtes dans l'infrastructure Cloud de chaque CP sélectionné. En prenant en compte la nature complexe du problème et le caractère dynamique et évolutif de l'environnement multiCloud, divers paramètres sont conjointement optimisés, incluant le taux d'acceptation, la performance et la Qualité de Service (QdS) des applications à héberger et du trafic entre ces applications à router dans les dispositifs IaaS, ainsi que les coûts induits d'approvisionnement des ressources nécessaires à cet effet.

Ce chapitre d'introduction commence par présenter certaines définitions et concepts de base permettant de mieux saisir le contexte de ce travail de recherche. Par la suite, les éléments liés à la problématique du VNE dans un environnement multiCloud seront exposés, suivis des objectifs de recherche visés dans cette thèse. Les principales contributions faisant l'originalité de ce travail de recherche seront ensuite mises en exergue, avant de clore ce chapitre par l'énoncé du plan de ce manuscrit.

1.1 Définitions et concepts de base

Cette section présente au lecteur les définitions et concepts de base utilisés tout au long de la thèse. À cet effet, nous commencerons d'abord par définir le concept de virtualisation de réseau, ainsi que celui du Cloud Computing et des modèles de service et de déploiement qu'il propose. Ensuite, nous décrirons les éléments techniques qui composent le réseau global multiCloud sous-jacent, suivi de ceux qui constituent une requête de réseau virtuel. Par la suite, les notions spécifiques liées aux exigences du requérant de service seront décrites, avant de terminer par la présentation des différentes phases qui hiérarchisent le processus global du VNE dans un environnement multiCloud.

1.1.1 Virtualisation

La virtualisation est une technologie de support clé dans le Cloud Computing permettant de faire abstraction des ressources matérielles d'un réseau (Murugesan et Bojanova, 2016). Elle se définit comme un ensemble de techniques offrant des moyens de regrouper, sur un seul support physique, plusieurs systèmes d'exploitation et ressources informatiques effectuant des tâches spécifiques, sous apparence qu'elles sont exécutées sur des supports physiques différents. La virtualisation permet ainsi d'offrir un environnement logique et indépendant du matériel physique, dans le but d'améliorer l'utilisation de ce dernier et de réduire des coûts d'investissements sur les dispositifs sous-jacents.

1.1.2 Cloud Computing

Avec le succès de l'Internet et le développement rapide des technologies de traitement et de stockage de données, les dernières décennies ont connu l'essor populaire du Cloud Computing (Armbrust *et al.*, 2010). Son concept s'appuyant notamment sur l'expansion des techniques de virtualisation, désigne un modèle informatique exploitant la mémoire et la puissance de calcul de plusieurs serveurs distants distribués géographiquement dans le monde. Ces derniers sont hébergés dans d'immenses centres de données reliés en réseau via Internet, dans le but de faciliter davantage l'accès, le stockage et le traitement de grands volumes de données. Cette technologie offre donc aux utilisateurs des moyens d'externaliser la gestion étendue de leur parc informatique (logiciels, plateformes, infrastructures), à des prestataires qui mettent à leur disposition, à la demande, diverses ressources informatiques configurables, proposées sous forme de services en ligne évolutifs pouvant rapidement être approvisionnés et libérés, et facturés à leur utilisation effective. Au fil des années, le Cloud Computing s'est vu largement adopté sur le marché des TIC, en particulier par les petites entreprises qui y voient des moyens de réaliser des économies considérables, en se dispensant de la gestion complexe et onéreuse d'une infrastructure physique.

Dans la suite, nous présentons les différents modèles de service et de déploiement proposés par la technologie du Cloud Computing.

1.1.2.1 Modèles de service

Le Cloud Computing se base sur un modèle d'affaires principalement axé sur trois (3) modèles standards de service (IBM, 2014; Murugesan et Bojanova, 2016), décrits ci-dessous et illustrés sur la Figure 1.1.

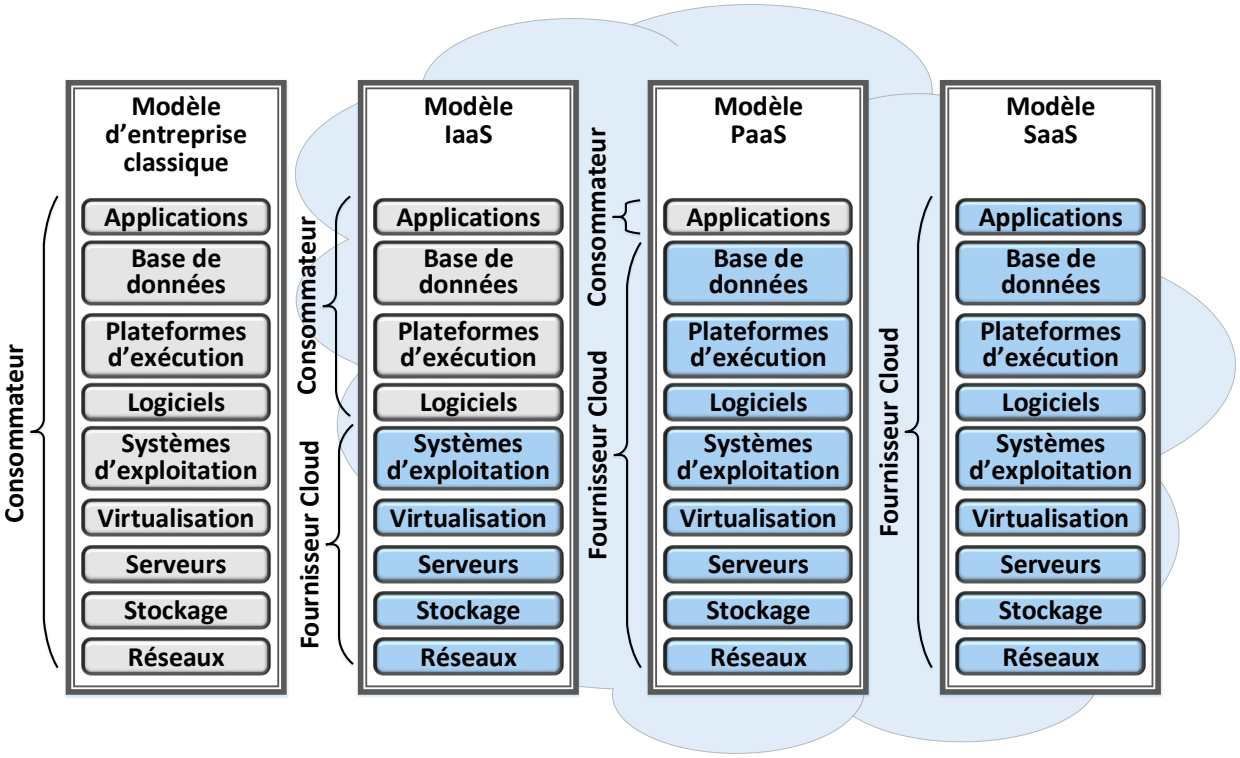

Figure 1.1 Modèles de service du Cloud Computing

— **Infrastructure-service ou *Infrastructure as a Service (IaaS)***, qui permet d'offrir aux consommateurs des ressources de la couche infrastructure. Dans ce modèle, le consommateur garde le contrôle sur le déploiement et l'exécution de ses applications, incluant les logiciels, les systèmes d'exploitation virtuels, les bases de données, alors que le fournisseur contrôle la gestion de l'infrastructure physique sous-jacente, incluant la virtualisation, les serveurs, le stockage et l'équipement réseau. L'Elastic Cloud Compute (EC2) d'Amazon Web Service (AWS) (Amazon, 2017), Microsoft Azure IaaS et Google Cloud sont des exemples populaires d'IaaS.

— **Plateforme-service ou *Platform as a Service (PaaS)***, qui permet d'offrir, en plus des services IaaS, un environnement de création et d'exécution d'applications, incluant des langages et outils de programmation. Dans ce modèle, le consommateur dématérialise donc le matériel, l'hébergement et la plateforme d'exécution de ses propres applications, tout en gardant le contrôle sur ces dernières. Microsoft Azure PaaS, Google App Engine et Red Hat OpenShift sont de PaaS connus.

— **Logiciel-service ou *Software as a Service (SaaS)***, qui est un modèle économique qui permet d'offrir au consommateur, en plus des services IaaS et PaaS, l'ensemble des applications hébergées sur le Cloud. Les services Office 365, Salesforce, Google Apps, Google Docs, Office Web Apps en sont des exemples.

1.1.2.2 Modèles de déploiement

Le déploiement des services Cloud peut se faire selon plusieurs modèles (IBM, 2014; Murugesan et Bojanova, 2016), tels que montré sur la Figure 1.2 :

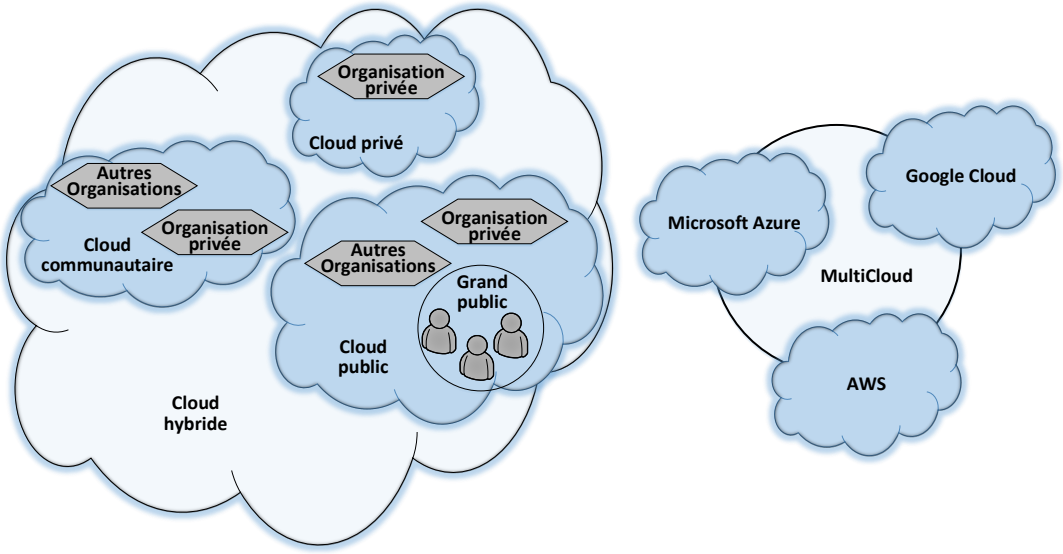

Figure 1.2 Modèles de déploiement du Cloud Computing

— **Cloud public**, où le Cloud est déployé à la disposition du grand public ou d'un groupe d'organisations, mais appartient cependant à un seul fournisseur. Un Cloud public est en général peu couteux pour les consommateurs, ces derniers n'investissant pas sur la gestion des services et des risques, mais en retour, ceux-ci n'ont pas un contrôle minutieux sur les données hébergées, ni sur les paramètres de réseau et de sécurité.

— **Cloud privé**, où le Cloud appartient à une organisation privée qui offre des services à ses consommateurs internes, cependant la gestion du Cloud peut revenir soit à ladite organisation ou à une tierce partie. Contrairement au Cloud public, un Cloud privé offre un plus haut degré de contrôle sur la fiabilité et la sécurité des ressources hébergées, ce qui nécessite de la part du consommateur un investissement sur l'infrastructure.

— **Cloud communautaire**, où le Cloud est exploité par un groupe d'organisations partageant les mêmes intérêts et idéologies, avec des préoccupations et exigences communes (mission, requis de sécurité, politique, juridiction, etc.). Un Cloud communautaire peut être géré par la communauté ou par une tierce partie, tout en offrant également d'un niveau de sécurité élevé.
— **Cloud hybride**, qui est une combinaison de plusieurs types de Cloud (public, privé ou communautaire) permettant de tirer avantage de chacun d'eux ou de compenser leurs limitations. Par exemple, en déployant un Cloud hybride, une organisation peut saisir les avantages d'un Cloud public et continuer à utiliser son propre centre de données pour sécuriser ses données sensibles. Les entités coopérant dans le Cloud hybride demeurent indépendantes, mais liées toutefois par l'entremise de technologies standards ou propriétaires dans le but de faciliter la portabilité et la sécurité des données et des applications entre les différents domaines Cloud.
— **MultiCloud**, où le consommateur peut sélectionner et utiliser différents services Cloud, proposés dans une architecture hétérogène unique par plusieurs fournisseurs de Cloud indépendants. Ce modèle de déploiement est entrain de gagner en popularité car permettant au consommateur d'obtenir les meilleures performances, en lui donnant plus de flexibilité quant aux choix des services déployés et en réduisant les risques de dépendance vis-à-vis d'un fournisseur unique. Le multiCloud se distingue du Cloud hybride, en ce sens qu'il fait référence à plusieurs services Cloud plutôt qu'à plusieurs modes de déploiement.

1.1.3 Eléments techniques du réseau IaaS sous-jacent

Dans le contexte de ce travail, nous considérons le modèle de service IaaS déployé en mode multiCloud. Cette section décrit, à un haut niveau d'abstraction, les éléments techniques qui composent respectivement les environnements multiCloud et intraCloud (Manvi et Shyam, 2014; Rabah *et al.*, 2015; Dietrich *et al.*, 2015).

1.1.3.1 Environnement multiCloud

La Figure 1.3 montre l'environnement multiCloud, composé de plusieurs réseaux d'infrastructure intraCloud publics appartenant chacun à un fournisseur indépendant, et reliés par des liaisons interCloud. Ces dernières matérialisent tout simplement un haut niveau de représentation du réseau dorsal interconnectant les différents domaines intraCloud. Un tel environnement fait intervenir plusieurs acteurs et éléments techniques en interaction, selon leurs rôles et propriétés :

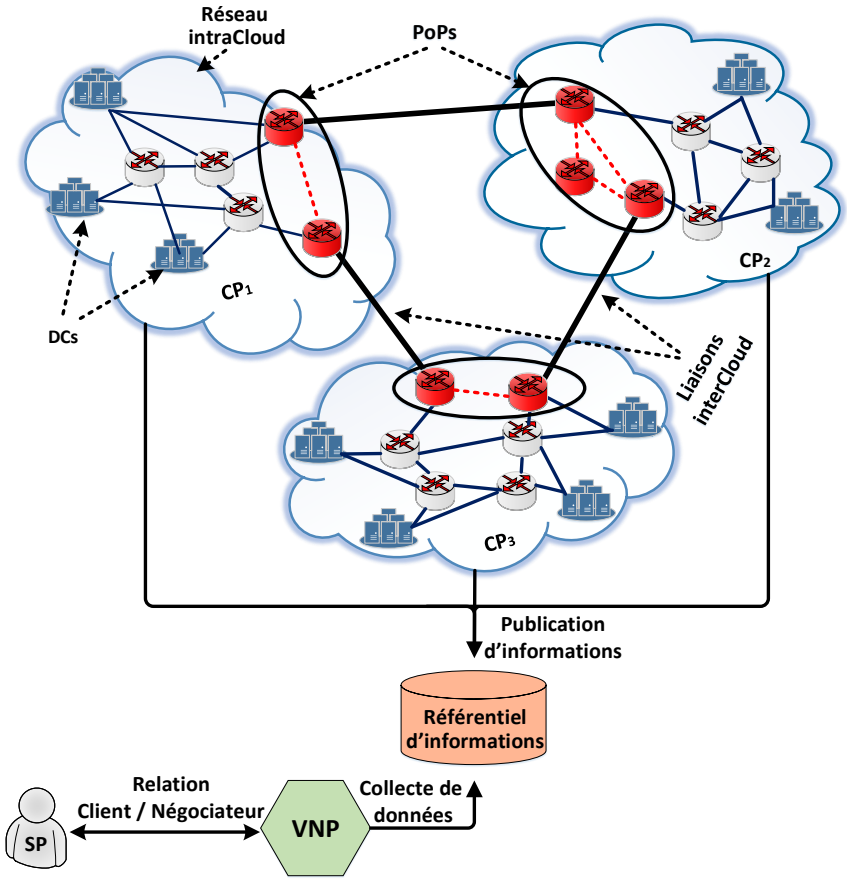

Figure 1.3 Environnement multiCloud

— **Fournisseur d'infrastructure** ou **Cloud Provider (CP)**, qui gère la virtualisation et la maintenance de ses ressources physiques, et se charge de déployer l'infrastructure et l'équipement réseau du Cloud virtualisé. Le CP offre ainsi plusieurs services IaaS, incluant des ressources virtuelles de calcul, de stockage et de réseau.

— **Fournisseur de service** ou **Service Provider (SP)**, qui loue les services des CPs pour héberger des applications personnalisées, qui seront offertes à leur tour également sous forme de services (le plus souvent sous le modèle SaaS) à d'autres utilisateurs finaux.

— **Fournisseur de réseau virtuel** ou **Virtual Network Provider (VNP)**, qui est en général un négociateur, faisant référence à un système virtuel de courtage de service, qui a une connaissance élargie du marché Iaas. Le VNP est alors mandaté par le SP pour collecter et agréger les données sur les multiples ressources virtuelles et services offerts par les CPs, afin d'opérer les meilleures sélections de CPs hôtes possibles pour le compte du SP.

— **Référentiel d'informations** ou **Information repository**, qui est une sorte d'entrepôt virtuel de données, alimenté par les référentiels d'informations locaux de chacun des domaines des CPs. Ces derniers y publient des informations sur les ressources et services virtuels qu'ils offrent, incluant les types de ressources, les systèmes d'exploitation, les plateformes de virtualisation, les zones géographiques couvertes, les coûts d'approvisionnement de ressources, ainsi que quelques statistiques sur l'état de leur réseau et des services garantis, tout en maintenant un certain niveau de confidentialité de leurs données internes.

— **Point de présence** ou **Point of Presence (PoP)**, qui désigne généralement le point d'accès à partir duquel les CPs établissent une connexion. Un PoP fait souvent référence à un ensemble de dispositifs de communication permettant d'accéder au domaine d'un CP, afin d'établir des liaisons interCloud (liaisons entre plusieurs CPs). Un PoP peut aussi permettre de transiter par le réseau d'un CP, afin de pouvoir établir des connexions entre des domaines qui ne sont pas reliés directement. Un CP peut gérer un ou plusieurs PoPs à différents endroits, chacun s'adressant à une base d'utilisateurs distincte. Le PoP peut aussi appartenir au CP, tout comme être géré par un tiers, généralement désigné comme fournisseur de réseaux de transit.

1.1.3.2 Environnement intraCloud

L'environnement intraCloud désigne le réseau d'infrastructure appartenant à un fournisseur de Cloud IaaS. Tel qu'illustré sur la Figure 1.3, il est représenté par une infrastructure distribuée, constituée de plusieurs centres de données ou Data Centers (DCs) situés à des régions géographiques différentes, et interconnectés à travers un réseau dorsal ou *backbone* (un vaste réseau fédérateur généralement semblable au réseau NSFNet (Amokrane *et al.*, 2015)). Les DCs, quant à eux, désignent d'immenses unités administratives regroupant des équipements informatiques et de puissants serveurs interconnectés en réseau, permettant d'héberger une vaste gamme d'applications.

1.1.4 Requête de réseau virtuel

Une requête de réseau virtuel ou Virtual Network Request (VNR) représente l'ensemble des ressources informatiques et réseautiques demandées par le SP dans le but de créer un réseau virtuel hétérogène capable de supporter ses diverses applications. Une VNR est généralement composée de machines virtuelles ou Virtual Machines (VMs) communicant entre elles à travers des liaisons virtuelles ou Virtual Links (VLs). Une VM se définit comme un environnement d'exécution qui, grâce aux techniques de virtualisation, reproduit les propriétés des

composantes informatiques d'un système hôte virtualisé. Ces composantes peuvent inclure le processeur, la mémoire, le disque dur, les cartes d'interface réseau, etc. L'application d'un SP peut être encapsulée dans une seule VM, ou dans plusieurs VMs en interaction. Une VL représente tout simplement le trafic échangé entre deux VMs qui partagent des informations sur le réseau. Dans ce travail, nous allons particulièrement nous intéresser au trafic inter-VMs, qui se distingue du trafic externe échangé entre les VMs et les clients finaux du SP.

1.1.5 Exigences du SP

Le SP soumet ses requêtes avec des exigences spécifiques pour les VMs et les VLs, auxquelles les CPs doivent répondre par l'entremise du VNP, selon une entente de service ou Service Level Agreement (SLA) bien définie (Fischer *et al.*, 2013; Manvi et Shyam, 2014). Nous définissons dans la suite les principales exigences et contraintes que peut soumettre le SP au VNP, dépendamment de ses objectifs et attentes.

1.1.5.1 Performance et qualité du service fourni

Le SP est particulièrement exigeant quant à la performance de ses applications hébergées. Au niveau du CP, un degré de performance satisfaisant pour le SP se traduit par la mise en œuvre de mécanismes appropriés et efficaces pour allouer les ressources nécessaires aux demandes du SP, et garantir leur disponibilité et leur facilité d'accès en continue, afin de répondre aux enjeux relatifs à la qualité de prestation de service.

La Qualité de Service (QdS) décrit la capacité à fournir un service conformément à des exigences bien définies. Son but est de garantir aux applications un traitement adéquat en fonction de leurs besoins en performance. Pour le SP, elle est perçue comme une satisfaction que ses requêtes sont correctement exécutées au regard de ses attentes de performance spécifiées dans ses requêtes. Pour le CP, la QdS est un moyen d'optimiser davantage l'utilisation de ses ressources et d'assurer des temps de réponses rapides aux requêtes, dans le but de répondre aux enjeux économiques liés au degré de satisfaction du SP.

1.1.5.2 Contraintes de localisation

En plus des exigences de QdS, le SP peut soumettre des contraintes quant à la localisation préférable des VMs à héberger, soit pour satisfaire des clients finaux ciblés dans une zone géographique bien définie, pour des questions de sécurité, de gouvernance et de territorialité, ou tout simplement pour éviter une concurrence dans un secteur visé. Les exigences de localisation peuvent aussi être exprimées sur les VLs, par exemple en spécifiant une dis-

tance ou un délai de communication maximal entre deux VMs, ce qui ajoute des contraintes supplémentaires quant à l'hébergement des VMs interconnectées au niveau des CPs.

1.1.5.3 Sécurité d'hébergement

Le SP a un regard exigeant quant à la sécurité de ses applications hébergées dans les infrastructures Cloud, notamment dans le cadre de modèles de déploiement publics. Un domaine Cloud étant par définition multi-tenant, le SP est particulièrement préoccupé par ses données confidentielles qui sont co-hébergées sur le même Cloud que celles de ses concurrents, sans ignorer les problèmes de cyber-attaques et risques de perte de données. De plus, l'accès aux données et aux applications se réalise par le biais de serveurs distants délocalisés et partagés, ce qui peut rendre la sécurité encore plus compromise. La sécurité devient alors un enjeu critique pour les CPs, qui se doivent de déployer tous les moyens pour mettre en œuvre des outils de cryptage informatique des données et des mécanismes sophistiqués d'authentification, garantissant une infrastructure fiable et des accès sécurisés aux applications. Ceci permettrait de pouvoir assurer pour le SP un niveau satisfaisant d'intégrité et de confidentialité de ses données hébergées, auxquelles il devrait de surcroît avoir un droit de regard.

1.1.5.4 Réduction des coûts d'approvisionnement

L'une des principales attentes du SP de la part du VNP est de lui sélectionner des CPs hôtes pouvant héberger ses applications selon les exigences définies précédemment, toutefois avec les moindres coûts possibles. Ces derniers peuvent inclure les coûts d'approvisionnement des ressources nécessaires pour héberger les VMs et faire router les VLs, mais également les coûts de services de transit et de communication interCloud (Dietrich *et al.*, 2015). L'ensemble des coûts d'approvisionnement peut également dépendre, par ailleurs, des types de VMs et VLs à satisfaire, du niveau de service et de sécurité souhaité, de la localisation choisie, de la durée d'hébergement des applications et de la fluctuation des demandes de services.

1.1.6 Processus global du VNE dans un environnement multiCloud

Dans le modèle IaaS, l'intégration des VNRs aux différentes infrastructures Cloud se traduit par une sélection stratégique et minutieuse des serveurs contenus chez les DCs des CPs et des chemins entre ces serveurs hôtes, respectivement pour l'hébergement des VMs demandées par le SP et le routage adéquat des VLs. Dans le modèle de déploiement multiCloud, cette procédure d'intégration des applications se fait hiérarchiquement en trois (3) principales étapes de prises de décision (Zhang *et al.*, 2016a), tel qu'illustré sur la Figure 1.4 :

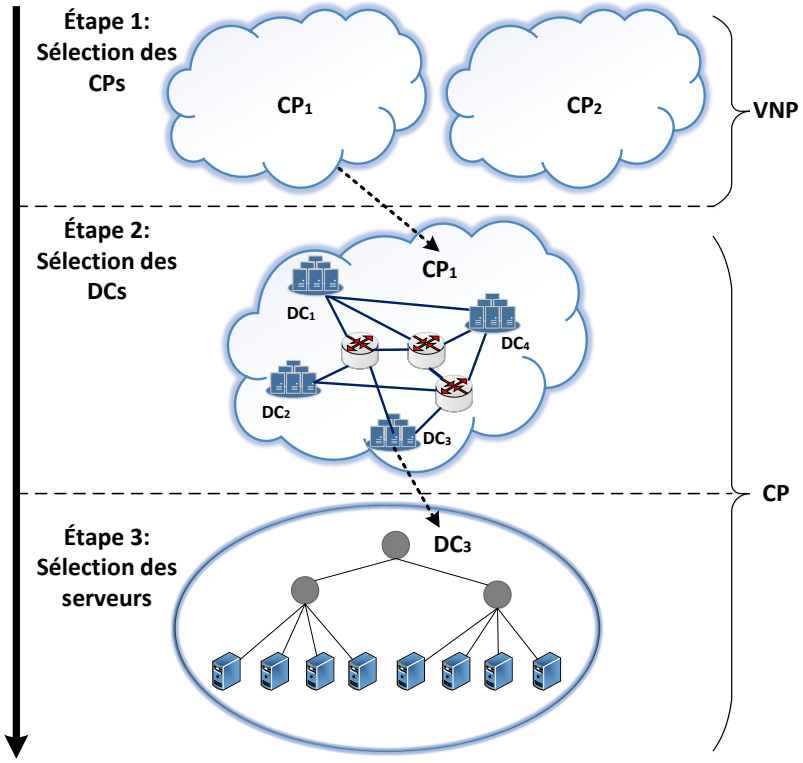

Figure 1.4 Étapes décisionnelles dans l'intégration des VNRs dans un environnement multi-Cloud

1. **Sélection des CPs** : La première étape consiste pour le VNP à sélectionner les meilleurs CPs capables de satisfaire les besoins et exigences du SP, tout en réduisant les coûts d'approvisionnement définis précédemment. Les multiples CPs se différencient en termes d'emplacement géographique, de disponibilité des ressources et des services, des types de VMs et VLs offerts et de leur prix d'hébergement. À l'issue du processus de sélection de CPs, les requêtes de services sont réparties entre chaque CP choisi, qui procède à leur tour aux étapes suivantes.
2. **Sélection des DCs** : Au niveau du CP sélectionné, un processus d'identification des DCs distribués géographiquement est ensuite initié, afin d'héberger les VMs aux emplacements désirés et faire router adéquatement le trafic inter-VMs entre les DCs. Le but pour le CP sera notamment de faciliter l'accès aux services demandés, tout en réduisant les coûts d'exploitation et d'opération dans les DCs, les coûts de trafic et les délais de communication inter-DCs, mais également d'optimiser l'utilisation de l'énergie pour ses équipements. Une fois les DCs sélectionnés, s'en suit la dernière étape de sélection des serveurs hôtes situés au niveau de chacun des DCs choisis.

3. **Sélection des serveurs** : Cette dernière étape consiste à sélectionner adéquatement les serveurs du DC auxquels assigner les VMs, et à router également le trafic inter-VMs. L'optimisation des coûts d'énergie et d'exploitation des ressources, de même que le respect des exigences en matière de performance et de QdS à fournir au SP, nécessitent à cette étape une utilisation efficace des composantes physiques serveur et réseau, qui peut par ailleurs être facilitée par des mécanismes performants de migration de VMs.

Dans ce processus de VNE multiCloud, les étapes de sélection faisant intervenir le VNP et le CP correspondent respectivement au partitionnement des VNRs à travers les CPs de l'environnement multiCloud et à l'hébergement des segments de VNRs dans les réseaux intraCloud sélectionnés. Les différentes sous-étapes que constituent ces deux phases, ainsi que celles d'une phase supplémentaire de validation du processus global, sont illustrées sur la Figure 1.5 et détaillées dans la suite.

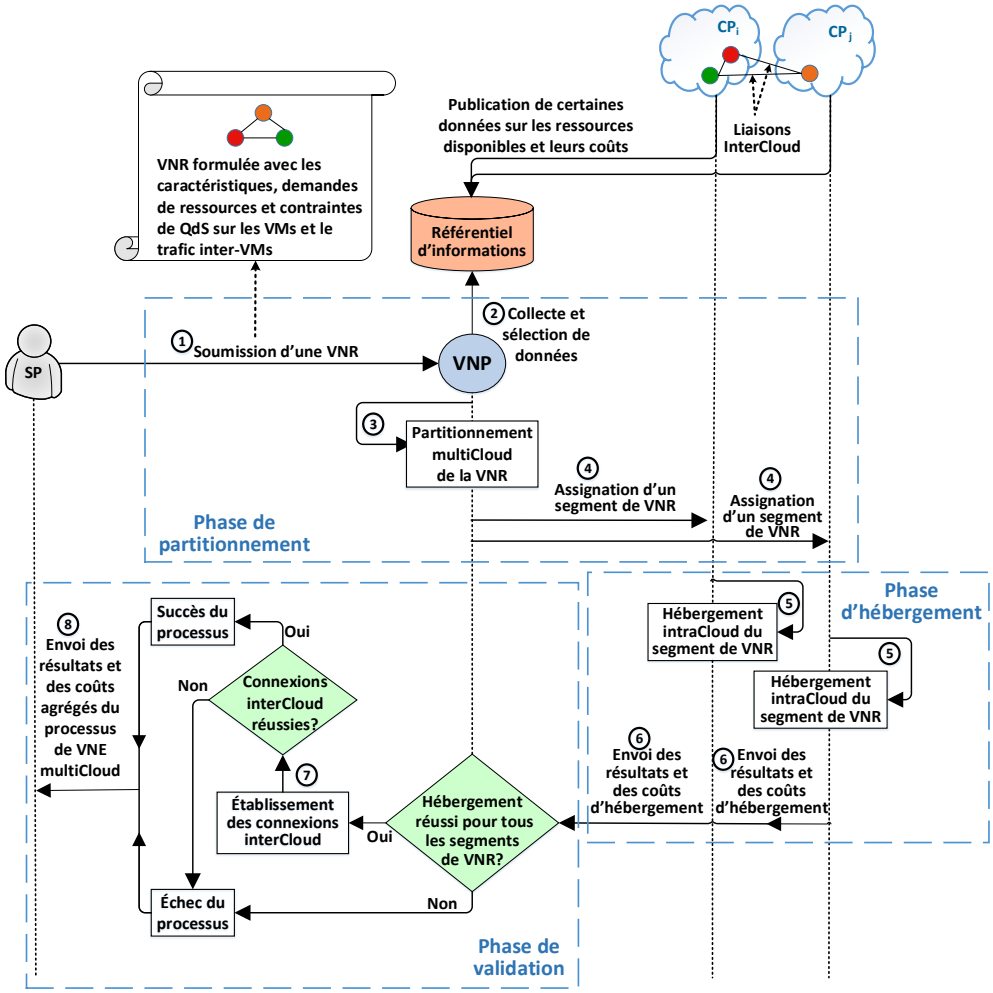

Figure 1.5 Processus global du VNE dans un environnement multiCloud

1.1.6.1 Phase de partitionnement des VNRs

La phase de partitionnement des VNRs est réalisée par le VNP pour le compte du SP suivant les étapes ci-dessous :

1. Le SP soumet au VNP une VNR (constituée d'un ensemble de VMs interconnectées par des VLs), accompagnée des besoins en ressources nécessaires pour la performance du service demandé (puissance de calcul, mémoire, stockage, bande passante, etc.), ainsi que des contraintes de QdS (localisation géographique, délai de communication inter-VMs maximal, niveau de sécurité, etc.).
2. Le VNP procède ensuite à la collection des données sur les ressources et services publiées par les CPs dans le référentiel d'informations, et fait une première sélection de CPs potentiellement candidats à répondre à certains critères de correspondance de base (type des VMs demandées, environnement de virtualisation, systèmes d'exploitation, empreinte géographique, type des VLs, etc.), généralement qualifiés d'attributs fonctionnels de la requête.
3. Dans la plupart des cas, pour pouvoir satisfaire toutes les exigences du SP, le VNP devra répartir, de manière stratégique et optimisée, la requête de service entre les éligibles CPs de l'étape de pré-sélection précédente. À cette fin, le VNP évalue les coûts d'approvisionnement les plus rentables, dépendamment de la disponibilité des ressources offertes par les CPs et de leur prix, tout en considérant les exigences de performance et de QdS spécifiées par le SP. Par la suite, le VNP procède au partitionnement de la VNR en plusieurs segments de requêtes.
4. Chacun des segments de VNRs résultant de la phase stratégique de partitionnement précédente est assigné à chacun des CPs sélectionnés, afin que ces derniers procèdent par la suite à la phase d'hébergement des segments de VNRs dans leur infrastructure intraCloud.

1.1.6.2 Phase d'hébergement des segments de VNRs

La phase d'hébergement des segments de VNRs est réalisée à l'échelle de l'infrastructure intraCloud par le CP, selon les étapes suivantes :

5. Chaque CP recevant un segment de VNR s'adonne à un processus d'hébergement des ressources informatiques et réseautiques demandées dans son infrastructure Cloud. Pour cela, le CP évalue d'abord la demande (ressources requises pour les VMs et les VLs ainsi que les contraintes de QdS associées au segment de VNR reçu), et procède à une collecte d'informations sur l'état de son infrastructure et des applications déjà

hébergées, par l'entremise d'une entité logique de coordonnateur de Cloud. Ce dernier permet d'agréger, à partir de chacun de ses DCs locaux et de son réseau dorsal, des statistiques sur l'utilisation des équipements, l'état des liens de communication, les quantités de ressources disponibles, la durée de vie restante des applications existantes, etc. Le CP opère ensuite un mécanisme optimisé d'assignation des VMs aux DCs et de routage des VLs sur des chemins appropriés entre les DCs hôtes à travers le réseau dorsal (voir Figure 1.3). L'objectif visé par le CP dans ce mécanisme est de réaliser des profits économiques et énergétiques, d'optimiser l'utilisation des ressources et les capacités de survie de l'infrastructure, et de réduire ainsi les coûts d'exploitation et d'opération sur son réseau intraCloud.

6. À l'issue de l'étape précédente, chaque CP renvoie au VNP les résultats d'hébergement, lui renseignant sur l'acceptation de la demande et des coûts induits, ou du rejet de la demande.

1.1.6.3 Phase de validation du processus

La phase de validation se fait dans les deux dernières étapes suivantes :

7. La VNP analyse chacun des résultats envoyés par les CPs. Dans le cas où un des CPs rejette la demande partielle qui lui a été assignée, le VNP considère l'ensemble du processus de VNE échoué. Dans le cas contraire, il procèdera à présent à la mise en connexion des différents segments de requêtes hébergés, via des réseaux d'accès et de transit, afin faire router le trafic inter-CPs sur des liaisons interCloud appropriées.
8. À l'issue de la validation de tout le processus du VNE multiCloud, l'ensemble des résultats de l'intégration de la VNR dans le multiCloud, incluant les coûts d'approvisionnement de ressources et services intraCloud et interCloud, est acheminé au niveau du SP.

Les concepts de base permettant de mieux cerner le contexte de la thèse étant définis, nous allons à présent analyser les éléments relatifs à la problématique du VNE dans un environnement multiCloud, plus particulièrement au niveau de la phase de partitionnement des requêtes où le VNP doit s'adonner à une sélection stratégique des CPs.

1.2 Éléments de la problématique

Bien que le modèle de service IaaS sous un mode déploiement multiCloud semble gagner du terrain sur le marché des TIC par les nombreux avantages qu'il offre au consommateur, il n'en

demeure pas moins que le processus d'intégration des VNRs auprès des multiples fournisseurs constitue un problème d'optimisation complexe à résoudre.

En effet, tel que défini précédemment, le processus global du VNE dans le réseau multiCloud sous-jacent nécessite deux grandes phases de résolution complexe, qui doivent de surcroît être traitées de manière séquentielle, ce qui ne permet pas de considérer ni de contrôler dans le processus, tous les facteurs du problème en interaction dans l'environnement.

De plus, les différents acteurs intervenant dans chacune des deux phases ont généralement des objectifs différents, voire conflictuels, dans l'optimisation de leurs intérêts économiques (Samuel *et al.*, 2013). Alors que les CPs s'intéressent généralement à maximiser les profits économiques et énergétiques, le SP est principalement préoccupé la minimisation des coûts d'approvisionnement. À cela s'ajoute le fait que le VNP n'ait pas un contrôle total sur les éléments économiques du marché multiCloud, à savoir les systèmes de facturation et modèles de tarification, qui peuvent être différents d'un CP à un autre. Il devient alors difficile pour le VNP de répondre, dans un tel contexte, aux objectifs économiques du SP, outre le niveau de performance de ses requêtes aux emplacements géographiques choisis et les aspects de QdS et de sécurité à satisfaire. Le VNP se retrouve confronté au défi principal de procéder à des décisions judicieuses et stratégiques de partitionnement des VNRs face à la grande panoplie hétérogène d'instances de ressources et de services déployés, disponibles différemment chez chaque CP et à des coûts d'approvisionnement variés.

Par ailleurs, l'un des plus grands défis relatifs au VNE dans le réseau multiCloud consiste à organiser les CPs dans un cadre d'approvisionnement de leurs services Cloud, sans passer outre leur autonomie locale (Dietrich *et al.*, 2015; Mano *et al.*, 2016). Chaque CP doit pouvoir intégrer des parties ou l'intégralité d'une requête conformément à ses politiques administratives internes, tout en maintenant une connectivité globale avec les autres CPs, par le biais des réseaux de transit. Toutefois, le VNP se heurte aux politiques de confidentialité des CPs et au manque d'interopérabilité entre eux, qui lui rendent la tâche difficile d'obtenir des solutions de configuration optimales, relatives au choix complexe des Clouds hôtes. En effet, la visibilité du VNP sur le réseau multiCloud est particulièrement essentielle pour l'efficacité de la stratégie de partitionnement des VNRs entre les différentes infrastructures. Cependant, les politiques restrictives des CPs entraînent que certaines informations internes sur leur réseau intraCloud, telles que le nombre d'instances de ressources disponibles, leur utilisation et leur capacité, la topologie détaillée du réseau sous-jacent et des PoPs, la connectivité au niveau des routeurs et des DCs, etc., ne sont pas divulguées au VNP. Par conséquent, les mécanismes de VNE existants et notamment appliqués dans le cadre d'un seul CP et qui se basent sur une connaissance complète du réseau substrat, ne sauraient être applicables dans le contexte du

multiCloud. Au niveau de la phase d'intégration intraCloud, les CPs peuvent donc générer des configurations idéales d'hébergement des segments de VNRs dans leur infrastructure, car ayant une vision complète et détaillée de leur réseau. Le VNP, quant à lui, doit procéder à la phase de partitionnement des VNRs avec un accès limité et une connaissance plutôt agrégée de l'environnement sous-jacent, ce qui rend difficile d'optimiser la phase de sélection des CPs.

D'autre part, la non-interopérabilité entre les CPs entraîne que les situations d'hébergement des segments d'une même VNR ne sont pas partagées entre les CPs auxquels ils sont assignés, ce qui fait qu'un CP procède au processus complet d'intégration d'un segment de VNR en particulier, sans aucune connaissance du résultat de l'intégration du reste de la requête. En cela résulte, du point de vue global du processus du VNE dans le multiCloud, une difficulté majeure pour le VNP de générer des solutions qui doivent au mieux répondre aux objectifs du SP, tout en se conformant aux politiques restrictives des CPs.

Par ailleurs, considérant la nature dynamique de l'environnement multiCloud (mise à jour périodique des informations contenues dans les référentiels, fluctuation des requêtes et des prix, etc.), il devient nécessaire de développer des mécanismes automatiques et adaptatifs d'intégration des VNRs, afin d'assurer la continuité du processus d'approvisionnement des services. Toutefois, le problème du VNE dans le multiCloud représente un défi majeur en matière d'allocation de ressources et de services dû à la nature NP-Difficile du problème (Fischer *et al.*, 2013; Zhang *et al.*, 2016a), autant pour la phase de partitionnement multiCloud que celle de l'hébergement intraCloud. En effet, dans le cas où le trafic inter-VMs n'est pas tenu en compte, le processus d'assignation des VMs seules aux différents CPs et DCs est un problème NP-Difficile, qui s'assimile généralement au problème de *Bin Packing* (Chowdhury *et al.*, 2012). Même si les VMs seraient déjà assignées et que seules les liaisons VLs restent à router, le problème reste NP-Difficile. Considérer alors la combinaison de l'assignation des VMs et le routage du trafic inter-VMs, ainsi que les contraintes associées, résulte en un problème encore plus difficile, généralement réduit au problème de *Multiway Separator* (Sanchis, 1989; Tao *et al.*, 1992).

À cet effet, des mécanismes utilisant des approches de résolution exactes, même si elles peuvent procurer des solutions optimales au problème, ne seraient pas appropriées. Ces dernières génèrent des temps de résolution extrêmement longs, notamment pour des configurations de VNRs ayant un nombre élevé de VMs. Il devient alors nécessaire de recourir à des approches approximatives et évolutives basées sur les (méta) heuristiques (Boussaïd *et al.*, 2013), qui permettraient de résoudre de manière efficace le problème et ce en un temps de calcul raisonnable.

Ces divers éléments de la problématique nous ont amenés à l'élaboration des questions de recherche suivantes :

- Serons-nous en mesure de proposer une stratégie efficace de partitionnement des VNRs dans l'environnement multiCloud, qui puisse répondre à toutes les attentes du SP, tout en se conformant aux politiques restrictives des CPs ?
- Comment modéliser mathématiquement le problème du partitionnement des VNRs, afin qu'une telle stratégie soit à même de considérer tous les besoins en performance et les contraintes de QdS conjointement liés aux VMs à assigner aux CPs et aux VLs à router sur les différents réseaux sous-jacents ?
- Partant de cette modélisation, pourrons-nous résoudre le problème du partitionnement des VNRs à l'aide d'un modèle de programmation mathématique, nous permettant de sélectionner efficacement les CPs, tout en prenant en compte la nature dynamique de l'environnement multiCloud ?
- Comment adopter et résoudre de manière adéquate une approche d'hébergement des requêtes dans les réseaux intraCloud des CPs, afin que le processus global d'intégration des VNRs dans l'environnement multiCloud puisse refléter la réalité et considérer au mieux possible toutes les interactions entre les différents éléments techniques et les acteurs intervenant dans l'environnement ?
- Saurons-nous proposer des mécanismes de résolution permettant de pallier la nature NP-Difficile du problème et de parvenir à un bon compromis entre qualité des solutions obtenues et temps de résolution ?
- Serons-nous en mesure de mettre en œuvre un processus global d'intégration des VNRs dans l'environnement multiCloud, qui puisse minimiser les coûts d'approvisionnement de ressources pour le SP face aux différents systèmes de tarification et de facturation des services, tout en considérant les objectifs d'optimisation des performances des applications et de maximisation des profits des CPs ?

Voici un ensemble de questions auxquelles nous tenterons de répondre par les objectifs de recherche présentés dans la section suivante.

1.3 Objectifs de recherche

Cette thèse vise principalement à concevoir un cadre hiérarchique d'intégration des requêtes de réseaux virtuels (VNRs) dans un environnement multiCloud offrant des services sous le modèle IaaS. Le cadre proposé inclut l'élaboration d'une première approche stratégique de partitionnement des VNRs parmi les multiples CPs de l'environnement, et l'adoption d'une

approche subséquente d'hébergement des segments de VNRs résultants dans le réseau d'infrastructure Cloud de chaque CP choisi. Tout en considérant la nature complexe du problème et le caractère dynamique de l'environnement multiCloud, l'objectif est de sélectionner efficacement les CPs selon les exigences et contraintes de QdS liées aux applications du SP à héberger, afin de maximiser le taux d'acceptation et la performance des demandes, tout en minimisant pour le SP les coûts induits d'approvisionnement de ressources et de services.

De manière plus spécifique, cette thèse vise à :

1. Modéliser le problème du partitionnement des VNRs à travers le réseau multiCloud dans le but d'optimiser la performance des VNRs et de maximiser le taux d'acceptation de celles-ci. La modélisation mathématique proposée prendra conjointement en compte les exigences en performance et contraintes de QdS spécifiées par le SP, autant sur les VMs à assigner aux CPs et que sur le trafic inter-VMs à router à travers le réseau sous-jacent, tout en considérant les politiques restrictives des CPs.
2. Concevoir un modèle de programmation mathématique permettant de résoudre le problème du partitionnement multiCloud des VNRs avec une méthode exacte.
3. Adopter et implémenter une approche adéquate de résolution du problème d'hébergement intraCloud, afin d'optimiser l'intégration des segments de VNRs dans les différentes infrastructures Cloud sélectionnées à la phase de partitionnement des VNRs.
4. Évaluer, selon plusieurs critères de performance, l'efficacité de la stratégie de partitionnement multiCloud proposée, en comparant notre modèle à d'autres de la littérature utilisant la même approche de résolution de la phase d'hébergement intraCloud.
5. Proposer et évaluer une méthode de résolution approximative basée sur les métaheuristiques, dans le but de résoudre des instances de grande taille du problème de partitionnement des VNRs et d'obtenir un bon compromis entre qualité des solutions générées et temps de calcul.
6. Étendre et évaluer, selon plusieurs facteurs de performance et d'économie, le modèle mathématique proposé pour le partitionnement des VNRs, afin d'également considérer la minimisation des coûts d'approvisionnement de ressources et de services pour le SP.
7. Proposer et évaluer une méthode de résolution approximative et adaptative basée sur une hybridation de différentes métaheuristiques, afin d'obtenir, en un temps de calcul raisonnable, des solutions satisfaisantes pour le nouveau modèle étendu de partitionnement des VNRs.

1.4 Principales contributions de la thèse et leur originalité

L'originalité de cette thèse réside dans la conception d'un nouveau cadre hiérarchique d'optimisation de l'intégration des VNRs dans un environnement multiCloud, un problème émergent qui n'a été que récemment adressé dans la littérature. Le cadre développé vise conjointement à optimiser le niveau de performance et de QdS des applications à héberger chez les multiples fournisseurs d'IaaS (CPs), et à minimiser les coûts d'approvisionnement des ressources informatiques et réseautiques pour le SP. La phase la plus délicate pour ce dernier étant d'opérer une démarche stratégique et rentable de sélection des CPs éligibles à héberger ses applications, les principales contributions de ce travail de recherche ont été apportées au niveau de la résolution du problème de partitionnement des VNRs à travers le réseau multiCloud. Toutefois, l'efficacité d'une telle stratégie ne saurait être évaluée sur un ensemble de critères de performance et d'économie, sans résoudre le problème subséquent d'hébergement des segments de VNRs dans les réseaux d'infrastructure des CPs sélectionnés durant la phase de partitionnement. Cette perspective nous a alors conduits à concevoir dans cette thèse, un cadre de résolution globale de l'ensemble du processus du VNE dans l'environnement multiCloud, dont les principales contributions peuvent être détaillées comme suit :

1. **Modélisation mathématique d'applications complexes :** Plutôt que de traditionnellement considérer un modèle IaaS n'offrant uniquement que des services d'hébergement de VMs, nous avons pu proposer un cadre d'intégration des requêtes dans un environnement multiCloud, permettant de considérer et de modéliser des applications complexes. Ces dernières sont formulées sous forme de VNRs, encapsulées dans plusieurs machines virtuelles (VMs), échangeant du trafic non négligeable entre elles à travers les liaisons virtuelles (VLs), ce qui ajoute plus de complexité au problème. Le cadre propose une modélisation des VNRs soumises par le SP, incluant leurs besoins en performance et contraintes en QdS et en localisation, et prend en compte tous les types de ressources que peut exiger une VM, tout en pondérant l'importance de chacune des composantes de ressources requises, dans l'optique d'optimiser les performances de la VM en fonction de l'application qu'elle encapsule. La modélisation proposée prend également en charge les applications sensibles au délai, formulées par le SP en spécifiant un délai maximal pour certains trafics inter-VMs à router.
2. **Modélisation mathématique de l'environnement multiCloud basée sur les politiques restrictives des CPs :** L'une des contributions de ce travail porte sur la modélisation de l'environnement multiCloud sous-jacent et des données collectées dans le référentiel d'informations, qui s'aligne aux politiques restrictives des CPs. En effet, comparé aux modèles existants qui assument soit une connaissance complète

et détaillée des réseaux sous-jacents, ou sinon utilisent néanmoins des informations confidentielles sur les CPs dans leur stratégie, nous avons pu dans notre modélisation étudier le niveau d'accès restreint du VNP sur l'environnement multiCloud, et concevoir une stratégie de partitionnement des VNRs basée sur les informations limitées et agrégées, publiées par les CPs. Par ailleurs, dans le but de fournir une meilleure efficacité de la stratégie de partitionnement proposée, nous avons enrichi la visibilité du VNP sur l'environnement multiCloud, en ajoutant dans la modélisation des données statistiques de performance sur les PoPs des CPs qui ne sont pas considérées confidentielles.

3. **Conception d'un nouveau modèle de programmation mathématique pour résoudre le problème de partitionnement multiCloud des VNRs :** Pour formaliser mathématiquement notre stratégie de sélection des CPs et résoudre le problème de partitionnement des VNRs, nous avons utilisé des techniques de programmation linéaire en nombres entiers ou Integer Linear Programming (ILP), dont les différents paramètres utilisés pour définir la fonction objectif et les contraintes linéaires du modèle, résultent de la modélisation préalable des VNRs et de l'environnement multiCloud sous-jacent. L'originalité dans le modèle ILP proposé repose sur la stratégie de sélection des CPs qu'il exécute, qui consiste en une optimisation conjointe de la performance et la QdS des VMs et des VLs aux emplacements géographiques de choix du SP, tout en prenant en compte la nature dynamique de l'environnement multiCloud et les mises à jour périodiques des données publiées dans le référentiel d'informations. De plus, le modèle de programmation définit le routage du trafic inter-VMs entre les CPs hôtes, en minimisant le nombre de réseaux de transit transit traversés, ce qui réduit considérablement le délai de communication inter-VMs tout en évitant les violations de contraintes de délai maximal imposées sur les VLs.

4. **Adoption d'un modèle de résolution multiobjectif du problème d'hébergement intraCloud des segments de VNRs :** Afin d'intégrer dans les infrastructures Cloud sélectionnées les segments de VNRs issus de la phase de partitionnement, nous avons adopté une approche multiobjectif basée sur un modèle ILP mixte ou Mixed-Integer Linear Programming (MILP) que nous avons adaptée au cadre de VNE multiCloud proposé. Le modèle multiobjectif vise précisément à minimiser, pour les CPs, les coûts d'utilisation de leurs ressources serveur et réseau, les coûts d'énergie, l'impact écologique et le délai de routage du trafic. L'implémentation d'une telle approche s'est avérée pertinente car permettant de refléter davantage la réalité du contexte multiCloud, mais également d'évaluer la stratégie de partitionnement appliquée selon plusieurs critères de performance, incluant le taux d'acceptation, le taux de partition-

nement, le délai de communication, le temps d'exécution et les violations de contraintes de QdS.

5. **Développement d'une approche de résolution heuristique permettant de pallier la nature NP-Difficile du problème du partitionnement des VNRs :** Une autre originalité de la thèse repose sur les mécanismes automatiques de résolution mis en œuvre. En effet, le modèle ILP proposé, résolu dans un premier temps avec une approche exacte, ne permet pas de traiter des instances de grande taille du problème car entraînant des temps de calcul exponentiellement longs dû à la nature complexe du problème. Pour pallier cette difficulté, nous avons élaboré une approche de résolution heuristique basée sur la Recheche Taboue ou Tabu Search (TS). L'approche proposée nous permet d'offrir un excellent compromis entre qualité des solutions générées par la métaheuristique et le temps d'exécution réduit à une complexité polynomiale.

6. **Optimisation conjointe de la performance et QdS des applications et des coûts d'approvisionnement pour le SP :** Dans le but de minimiser les coûts d'approvisionnement de ressources et de services pour le SP, une extension du modèle ILP de partitionnement est proposée, afin d'y intégrer les objectifs économiques du SP. Cette optimisation conjointe dans la stratégie de sélection des CPs nous permet d'apporter une nouveauté dans la littérature, en offrant un modèle de partitionnement multiCloud des VNRs regroupant divers facteurs de performance, de QdS et d'économie à optimiser pour le SP, ce qui lui laisse le choix de sélectionner les CPs selon ses objectifs de préférence.

7. **Développement d'une approche heuristique hybride pour la résolution du nouveau modèle de partitionnement des VNRs :** Une autre contribution de la thèse constitue un apport autant dans l'élaboration de mécanismes avancés de résolution du problème de partitionnement multiCloud des VNRs, mais également dans l'adaptation des métaheuristiques aux problèmes combinatoires complexes. En effet, pour résoudre plus efficacement et en un temps de calcul réduit le nouveau modèle stratégique de partitionnement des VNRs, nous avons développé une approche hybride basée sur une combinaison intégrative des métaheuristiques de colonies de fourmis ou Ant Colony Optimization (ACO), choisie pour ses capacités d'adaptation, et de TS, connue pour son efficacité en matière de recherche locale. Cette contribution apporte une originalité par l'utilisation, peu commune, des techniques d'hybridation de différentes métaheuristiques, permettant de tirer profit des capacités de chacune des méthodes combinées et d'obtenir des solutions davantage de meilleure qualité.

8. **Réduction considérable du temps d'exécution des approches heuristiques par la définition de fonctions de calcul avancées :** La dernière contribution de notre travail, est relative à la méthode d'évaluation des configurations de solutions générées par les approches heuristiques proposées. En effet, au lieu de communément évaluer entièrement et itérativement une solution à chaque modification de la configuration, ce qui est très couteux en temps de calcul, l'idée nous est venue de définir des fonctions de calcul de gains et de variations de coût, afin d'accélérer considérablement le temps d'exécution dans l'exploration de l'espace de recherche. À cet effet, des fonctions de calcul de gains sont établies pour l'approche basée sur TS, afin d'uniquement évaluer la différence entre le coût d'une solution actuelle et celui d'une solution voisine. Aussi, pour l'approche hybride, dans les phases de construction probabiliste d'une configuration de solution par l'algorithme d'ACO, des fonctions de calcul de variations de coût sont définies pour uniquement évaluer la différence de coût générée par l'ajout d'un nouveau composant à la configuration partielle d'une solution actuelle. Cette technique est plus compliquée à implémenter, mais elle nous permet de réduire considérablement l'ordre de complexité du temps de calcul de $O(n^2)$ à $O(n)$.

Du point de vue global de l'ensemble du processus de VNE dans un environnement multi-Cloud, le cadre proposé dans cette thèse pour l'intégration des VNRs dans les infrastructures Cloud, incluant la modélisation du problème et les mécanismes avancés de résolution, permettra d'offrir au SP un modèle décisionnel varié, dans une perspective de sélection stratégique et optimisée des multiples ressources et services Cloud les mieux adaptés en fonction de ses intérêts et de ses objectifs.

1.5 Plan du mémoire

Après avoir défini les notions et concepts de base, décrit les éléments de la problématique, énoncé les objectifs de recherche et souligné les principales contributions de cette thèse, la suite de ce manuscrit se présente comme suit : le chapitre 2 passe en revue les principaux travaux relatifs à l'optimisation de l'intégration des VNRs dans un environnement multiCloud. Plus particulièrement, il s'agira de faire état de la littérature sur le problème du VNE dans un tel contexte, et sur les approches généralement utilisées pour la résolution des problèmes d'optimisation d'une telle classe de complexité. Une analyse des limites et failles des travaux existants par rapport aux éléments de la problématique adressés sera également effectuée. Le chapitre 3 décrit les démarches de l'ensemble du travail de recherche réalisé, en mettant en évidence la relation entre les objectifs de recherche énoncés à la Section 1.3 et les articles scientifiques résultant de cette thèse.

Le chapitre 4 présente le texte intégral d'un article scientifique intitulé « *A QoS-based splitting strategy for a resource embedding across multiple cloud providers* », accepté pour publication dans le journal *IEEE Transactions on Services Computing*. L'article propose une stratégie de partitionnement des VNRs dans un environnement multiCloud, et définit à cet effet un cadre hiérarchique de VNE dans un tel contexte. Le cadre inclut notamment une approche de résolution de la phase subséquente d'hébergement intraCloud, implémentée dans le but d'évaluer et de valider la stratégie de partitionnement proposée. Cette dernière repose particulièrement sur un modèle mathématique ILP, qui vise à maximiser le taux d'acceptation et les performances des VNRs, en fonction des critères de choix du SP et des contraintes de QdS spécifiées sur les VMs et sur les VLs constituant les différentes applications à héberger dans les infrastructures Cloud. Le modèle mathématique est résolu avec un algorithme basé sur la méthode exacte, et l'efficacité de la stratégie de partitionnement proposée est démontrée dans l'article selon plusieurs critères de performance et de QdS.

Le chapitre 5 présente le texte intégral d'un article intitulé « *A Tabu Search Approach for a Virtual Networks Splitting Strategy Across Multiple Cloud Providers* », accepté pour publication dans la revue *International Journal of Metaheuristics*. L'article propose une méthode de résolution approximative adaptée à la métaheuristique de TS, afin de résoudre le problème de partitionnement des VNRs dans un environnement multiCloud en un temps de calcul polynomial. Une telle approche a été élaborée dans le but de pallier la nature NP-Difficile du problème, et de pouvoir ainsi générer des solutions pour de grandes instances du problème que la méthode exacte, proposée dans notre premier article, ne pouvait pas solutionner en un temps raisonnable. Les résultats présentés dans ce deuxième article montrent que l'approche heuristique proposée est capable de générer des solutions (presque) optimales, et ce en un temps de calcul considérablement réduit. Les performances observées permettent à l'algorithme heuristique d'être également aussi efficace que la méthode exacte pour prendre les meilleures décisions stratégiques de partitionnement des VNRs, en fonction des mêmes mesures de performance que celles considérées dans le premier article de notre travail.

Le chapitre 6 présente le texte intégral d'un article intitulé « *An efficient approach based on Ant Colony Optimization and Tabu Search for a resource splitting across multiple cloud providers* », soumis dans le journal *IEEE Transactions on Cloud Computing*. L'article propose une extension du modèle mathématique présenté dans nos deux premiers articles, dans le but d'offrir un cadre de VNE dans un environnement multiCloud, qui optimise conjointement les aspects de performance et de QdS des VNRs, mais également les coûts d'approvisionnement de ressources et de services Cloud du SP. Dans une perspective d'obtenir en un temps de calcul réduit des solutions davantage de meilleure qualité pour le nouveau modèle stratégique de partitionnement des VNRs, ce troisième article propose une approche de résolution basée

sur une hybridation des métaheuristiques d'ACO et de TS. L'efficacité et les performances satisfaisantes de l'approche hybride sont mises en évidence dans l'article, de même que la pertinence de l'optimisation conjointe proposée, autant du côté du SP que de celui des CPs.

Un analyse générale des démarches méthodologiques effectuées et des résultats obtenus dans le cadre cette recherche est ensuite présentée dans le chapitre 7. Pour finir, le chapitre 8 vient clore cette thèse, en présentant une synthèse des travaux de recherche réalisés et en énonçant, entre autres, les différentes limitations et les potentielles orientations de recherche pouvant étendre davantage notre travail dans le futur.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre présente les principaux travaux qui ont adressé le problème du Virtual Network Embedding (VNE) dans le cadre de la virtualisation et du Cloud Computing, et plus particulièrement avec le modèle de service IaaS. Une analyse sommaire du problème est d'abord effectuée. Par la suite, le chapitre passe en revue les travaux majeurs qui ont été réalisés, respectivement dans le cadre du VNE dans un environnement multiCloud et intraCloud. Les approches de résolution généralement adoptées pour solutionner un tel problème sont ensuite examinées, avant de terminer le chapitre par une analyse critique des travaux faisant état de cette revue de littérature.

2.1 Analyse sommaire du problème

Le VNE représente un problème complexe d'optimisation dans le domaine de la virtualisation (Fischer *et al.*, 2013; Zhang *et al.*, 2016a). Dans un environnement Cloud offrant des services sous le modèle Infrastructure-Service (IaaS), le problème fait référence au défi majeur de l'intégration efficace et automatique des requêtes de réseaux virtuels (VNRs) dans une infrastructure sous-jacente, virtualisée, dynamique et partagée, suivant un ensemble d'exigences et de contraintes liées aux demandes à satisfaire.

Dans la littérature, le problème du VNE est communément traité dans le cadre d'un environnement à un seul Cloud, où il est question pour le fournisseur de Cloud (CP) de procéder au processus du VNE, sur la base d'une connaissance complète de son réseau interne, dans le soucis d'optimiser l'utilisation de ses ressources informatiques et réseautiques, l'utilisation de l'énergie et les profits à générer. Le problème du VNE n'a été que très peu abordé dans le cadre d'un environnement multiCloud, un contexte qui ajoute un niveau de difficulté supplémentaire au problème, car nécessitant une phase préalable de partitionnement des VNRs entre les différents CPs. Cette étape incombe la lourde préoccupation pour le fournisseur de service (SP), de procéder à une optimisation stratégique de sélection des Clouds pour partitionner les requêtes, tout en se heurtant aux politiques restrictives des CPs, à leur manque d'interopérabilité et à leurs variétés d'offres de ressources et de services sous des modèles de tarification différents d'un CP à un autre.

Par ailleurs, nombreux sont les travaux portant sur le problème du VNE dans le contexte du Cloud Computing qui ne considèrent que des requêtes de services constituées uniquement de machines virtuelles (VMs) . Dans ce cas-ci, le problème est un peu plus simple car gé-

néralement assimilé au problème de *Bin Packing* (Chowdhury *et al.*, 2012), où il s'agit de définir un processus optimisé d'assignation de VMs aux serveurs logés dans les centres de données (DCs) des CPs. Cependant, les diverses gammes d'applications nécessitent souvent des requêtes complexes de services IaaS, soumises sous forme de VNRs, où les applications sont souvent encapsulées dans plusieurs VMs échangeant du trafic entre elles à travers les liaisons virtuelles (VLs). Dans cette circonstance, le processus de VNE revient alors à assigner également le trafic inter-VMs à des chemins adéquats entre les serveurs hôtes des VMs, ce qui est déjà intrinsèquement complexe. Dès lors, combiner l'assignation des VMs et le routage du trafic inter-VMs, accentue davantage le niveau de difficulté du problème, souvent assimilé au problème de *Multiway Separator* (Sanchis, 1989; Tao *et al.*, 1992). Par conséquent, plusieurs travaux dans ce contexte d'applications complexes à intégrer, optent communément à résoudre au préalable le problème d'assignation des VMs, pour se consacrer par la suite au problème de placement des VLs à travers le réseau sous-jacent, en utilisant par exemple des algorithmes de plus court chemin. Cependant, procéder ainsi de manière séquentielle, sans tenir compte des relations mutuelles entre les VMs et les VLs, peut aboutir à des performances insatisfaisantes. De ce fait, le développement de mécanismes d'optimisation simultanée de ces deux entités constituant les VNRs, est alors préférable pour une meilleure efficacité du processus de VNE.

À côté de ces éléments de difficulté du VNE, il est principalement à noter que le problème est NP-Difficile à plusieurs niveaux du processus global (Fischer *et al.*, 2013; Chowdhury *et al.*, 2012), ce qui impose, à la plupart des travaux de la littérature, la nécessité de développer des approches de résolution heuristiques pour être à même d'obtenir des configurations de VNE satisfaisantes, en un temps polynomial .

2.2 VNE dans un environnement multiCloud

Le problème du VNE dans un environnement multiCloud n'a été que récemment et peu adressé dans la littérature, comparé à celui à Cloud unique qui a été largement traité. Par conséquent, la plupart des travaux de recherche dans le domaine s'intéressent à développer des stratégies relatives à la phase de sélection des CPs. Par la suite, une méthode de VNE intraCloud est généralement adoptée, afin de résoudre la phase subséquente d'hébergement des applications partitionnées dans les infrastructures intraCloud, et de valider l'efficacité de la stratégie élaborée selon les attentes du SP. Cependant, tel que annoncé précédemment, la principale difficulté pour le fournisseur de réseau virtuel (VNP) dans la mise en œuvre d'une telle stratégie, est notamment liée aux politiques de confidentialité des CPs. Dans la suite, nous présentons les travaux qui ont adressé la question des politiques restrictives dans leur

stratégie de partitionnement, avant de discuter des différentes propositions sur le sujet du VNE multiCloud, ou dans des domaines connexes, selon plusieurs objectifs d'optimisation.

2.2.1 Partitionnement selon les politiques restrictives des CPs

La visibilité du VNP sur les données publiées dans l'environnement multiCloud est essentiellement critique pour l'efficacité d'une stratégie de partitionnement. Par conséquent, nombreux sont les travaux qui assument des données confidentielles des CPs comme entrées dans leur modèle de partitionnement, ou voire même assumer une totale connaissance des réseaux intraCloud sous-jacents, ce qui ne serait pas réaliste. De ce fait, en respect aux politiques qui limitent l'accès à certaines données confidentielles liées à la topologie et aux instances de ressources internes des CPs, certains travaux ont analysé le niveau d'accès du VNP aux informations de l'environnement multiCloud, avant de procéder à la phase de partitionnement. Les auteurs (Dietrich *et al.*, 2015), dans le but de minimiser les coûts d'approvisionnement des ressources pour le SP, ont étudié la sous-optimalité de leur modèle de VNE multiCloud qui est basé sur une divulgation limitée des informations disponibles, comparé à la situation où toutes les informations sur l'ensemble des réseaux Cloud seraient accessibles. Pour enrichir davantage la visibilité du VNP, une étude sur les données non confidentielles sur les PoPs des CPs est réalisée, afin d'utiliser la localisation et les coûts divulgués sur les nœuds et liens d'appairage, et d'améliorer ainsi leur technique de partitionnement. Dans la même lancée, (Mano *et al.*, 2016) ont également proposé une méthode d'optimisation limitée aux informations privées des CPs, afin d'héberger des VNRs sur plusieurs domaines Cloud en compétition. Leur modèle utilise des techniques de calcul multipartite sécurisé ou Multi-Party Computation (MPC), pour masquer des valeurs sensibles en entrée et préserver ainsi la confidentialité des CPs.

Par ailleurs d'autres travaux considèrent plutôt des informations agrégées, pouvant être divulguées au VNP sans entraver les politiques de confidentialité des CPs. Ces informations peuvent inclure, en plus d'une vue de haut niveau et simplifiée des réseaux PoPs (Dietrich *et al.*, 2015), des statistiques de trafic et certaines garanties de performance (Gong *et al.*, 2016), telles que les délais moyens sur un lien, la capacité maximale ou minimale d'une ressource disponible chez un CP(Chaisiri *et al.*, 2012; Li *et al.*, 2016), et ce sans en révéler les détails sur la répartition et l'utilisation dans leur réseau intraCloud.

2.2.2 Objectifs d'optimisation du VNE multiCloud

Les objectifs d'optimisation dans les modèles de VNE multiCloud proposés dans la littérature peuvent varier selon l'acteur intéressé. En effet, les stratégies de partitionnement développées

pour les intérêts du SP, visent généralement à optimiser la performance et la QdS de ses applications, ainsi que les coûts d'approvisionnement des ressources et services nécessaires, tandis que du côté du CP, les travaux s'orientent plus à définir des modèles de tarification efficaces pour maximiser leurs profits, dans un contexte compétitif de théorie de jeu.

2.2.2.1 Coûts d'approvisionnement

La majorité des travaux de la littérature sur le sujet du VNE multiCloud visent principalement à minimiser les coûts d'approvisionnement pour le SP. (Houidi *et al.*, 2011) sont parmi les premiers auteurs travaillant dans ce domaine de recherche, en proposant une approche qui compare des méthodes exacte et heuristique, en termes d'efficacité de leur modèle de partitionnement dans la réduction des coûts d'approvisionnement pour le SP. Pour résoudre la phase d'hébergement intraCloud, un algorithme exact est proposé dans le but d'augmenter le taux d'acceptation des VNRs, tout en réduisant les coûts d'exploitation de ressources pour les CPs. (Dietrich *et al.*, 2015) et (Gong *et al.*, 2016), dans leur étude de sous-optimalité de leur modèle de partitionnement, ont proposé une stratégie minimisant les coûts d'hébergement des VNRs, en assignant simultanément les VMs et les VLs, respectivement aux nœuds et liens d'appairage des PoPs selon les coûts qui y sont publiés. Leurs résultats montrent, selon plusieurs critères de performance, des distances faibles par rapport au modèle où toutes les données seraient accessibles. Dans les deux propositions, un modèle exact est aussi utilisée pour résoudre la phase d'intégration intraCloud, dans le but de minimiser l'utilisation des ressources CPU et de la bande passante pour le CP. (Gong *et al.*, 2016) considèrent de plus les capacités de survie des liens inter-domaines, ce qui améliore davantage le taux d'acceptation comparé à (Dietrich *et al.*, 2015). (Chaisiri *et al.*, 2012) et (Ran *et al.*, 2016) ont proposé un cadre d'assignation de VMs dans un environnement multiCloud, en tenant en compte l'incertitude de la demande des SPs et l'évolution du prix dans le temps, à différents stades de la réservation sur demande de ressources. Leur modèle utilise la programmation stochastique et les méthodes d'approximation par échantillonnage, afin de minimiser les coûts d'approvisionnement des ressources sur une période donnée. (Chaisiri *et al.*, 2012) utilisent en plus des méthodes de décomposition de Benders, afin de décomposer le problème d'optimisation en plusieurs sous-problèmes plus petits pouvant être résolus indépendamment et parallèlement, permettant ainsi de réduire le temps nécessaire pour obtenir une solution de l'algorithme proposé. Cependant, seulement des entités VMs sont optimisées pour les deux propositions, sans considération du trafic qui pourrait exister entre les VMs. Le modèle de (Mano *et al.*, 2016), se basant sur des opérations minimales de MPC, met en œuvre un mécanisme de partitionnement permettant de réduire les dépenses d'hébergement des VNRs chez les CPs. Dans leur modèle, chaque CP identifie des segments d'une VNR demandée pouvant

être hébergés dans son réseau intraCloud. De ce fait, le SP utilise uniquement l'ordre de prix estimé par l'algorithme de tri MPC pour chacun des segments, afin d'optimiser la sélection des CPs à qui les leur assigner et minimiser le coût total d'approvisionnement.

2.2.2.2 Performance et QdS

Certains travaux sur le sujet s'intéressent particulièrement à optimiser la performance et la QdS dans leur modèle de partitionnement. (Leivadeas *et al.*, 2013) ont proposé un cadre hiérarchique de VNE multiCloud, qui inclut un modèle de partitionnement des VNRs basé sur la rareté et l'utilisation moyenne des ressources disponibles chez chaque CP dans l'environnement. L'approvisionnement des ressources interCloud est défini et pénalisé en fonction du nombre de réseaux de transit utilisés sur les liens de communication. Par la suite, leur modèle utilise celui défini par (Papagianni *et al.*, 2013), pour intégrer les segments de VNRs résultants dans chacun des réseaux intraCloud sélectionnés. L'approche proposée par les auteurs génère une performance satisfaisante, notamment en terme d'évolutivité. Cependant, les coûts relatifs à la disponibilité des ressources sont définis en fonction du nombre de DCs et de liens physiques détenus chez les CPs, et de l'utilisation moyenne de ceux-ci, ce qui ne reflète pas nécessairement les performances de l'infrastructure d'un CP. (Li *et al.*, 2016) ont proposé un cadre d'intégration conjointe des VLs intraCloud et interCloud, dans le but d'optimiser l'utilisation des ressources et augmenter le taux d'acceptation chez les CPs. (Mechtri *et al.*, 2017) ont récemment proposé un modèle générique de VNE dans un environnement multiCloud basé sur une métrique de distance binaire. Cette distance mesure la proximité entre les ressources demandées (CPU, mémoire, stockage) et celles sélectionnées, ainsi que la proximité entre le délai sur un chemin sous-jacent et celui exigé sur une VL, que le chemin ne doit pas dépasser. De ce fait, la distance mesurée permet d'éliminer les solutions irréalisables en termes de violation des ressources minimales requises et des délais maximaux tolérés. Cependant, l'approche proposée ne prend pas en compte les ressources requises sur les VLs en termes de bande passante.

2.2.2.3 Modèles de tarification

Certains travaux ont adressé le problème de la tarification des ressources et des service au sein de l'environnement multiCloud (Samuel *et al.*, 2013; Ardagna *et al.*, 2017; Cardellini *et al.*, 2018), en proposant des modèles d'optimisation, le plus souvent dynamiques et basés sur la théorie des jeux, afin de gérer la compétition entre les CPs, ou alors les intérêts conflictuels entre les CPs et le SP. En effet, les auteurs abordent souvent le conflit qui existe entre les différentes entités intervenantes, essayant chacune d'optimiser leurs fonctions d'utilité. Les

CPs s'efforcent d'une part à optimiser l'allocation de leurs équipements, tout en s'intéressant, de manière concurrente, à recevoir la plus grande part possible du marché Cloud qui maximise leurs revenus. Le SP, quant à lui, est intéressé par la satisfaction de ses requêtes, tout en minimisant ses dépenses. (Samuel *et al.*, 2013) ont, par exemple, introduit un protocole décentralisé et distribué qui coordonne les CPs participant dans le multiCloud, dans le but de maximiser leurs revenus à travers des mécanismes compétitifs de tarification basés sur des enchères répétitives à chaque étape du processus du VNE. (Ardagna *et al.*, 2017) ont proposé un modèle basé sur la théorie des jeux, afin de résoudre la situation conflictuelle entre les CPs et les SPs. Leur modèle fait intervenir plusieurs SPs, qui compétitionnent entre eux pour l'utilisation des ressources offertes par les multiples CPs. Leur but est de minimiser les coûts d'approvisionnement des ressources louées à partir des CPs, qui à leur tour essayent de maximiser leurs revenus, en évitant notamment des pénalités de coût pour les échecs de traitement de requêtes. La situation conflictuelle est alors modélisée comme un problème d'équilibrage de Nash (D'Oro *et al.*, 2017), où les fonctions d'utilité de chacune des entités dépendent de la stratégie adoptée par les différents acteurs de l'environnement. (Cardellini *et al.*, 2018) ont également utilisé une approche de théorie de jeu, afin de mettre en œuvre, pour le CP, des modèles de tarification dynamique lui permettant de gérer l'approvisionnement sur demande de ressources de plusieurs SPs en compétition.

2.3 VNE dans un environnement intraCloud

Le problème du VNE dans le cadre d'un seul fournisseur de réseau Cloud correspond au problème d'hébergement intraCloud des demandes, et a été largement traité dans la littérature, sous différentes angles de recherche. En effet, dans ce contexte, le CP peut profiter de la connaissance complète et détaillée qu'il a de son réseau d'infrastructure interne, afin de pouvoir optimiser différents critères à son profit. Nombreux sont les travaux qui ne s'intéressent qu'à la phase de sélection des serveurs, telle que présentée à la Section 1.1.6, qui consiste généralement à optimiser l'utilisation de l'équipement physique logé au sein d'un DC. Cependant, dans cette revue de littérature, nous nous intéresserons particulièrement aux travaux qui incluent un niveau d'implication encore plus haut de la hiérarchie de déploiement, à savoir la sélection des DCs au sein du réseau intraCloud. Quelques propositions pertinentes sont alors présentées dans la suite, dépendamment des objectifs d'optimisation visés par le CP.

2.3.1 Optimisation de l'utilisation des ressources et des capacités de survie du réseau Cloud

Les CPs sont particulièrement préoccupés par l'optimisation de l'utilisation de leur équipement et les questions de capacités de survie de leur infrastructure. (Ayoubi *et al.*, 2015) ont proposé un cadre de VNE permettant de traiter des services Cloud à multidiffusion. Un modèle mathématique est utilisé pour formuler le problème, dans le but d'optimiser le taux d'acceptation et l'équilibrage des charges du réseau. Les auteurs ont ensuite proposé une extension du travail (Ayoubi *et al.*, 2016), afin d'intégrer les aspects dynamiques liés à la capacité de survie de l'équipement physique et à la variation des demandes de ressources. Pour cela, des mécanismes de migration et de re-localisation des VMs, s'adaptant à la disponibilité et à la fiabilité du réseau Cloud, sont alors mis en œuvre. Dans la même lancée, (Shahriar *et al.*, 2017) ont également adressé les questions de capacité de survie du réseau, en proposant un modèle proactif de VNE. Ce dernier est basé sur des stratégies de reconfiguration et de réacheminement de trafic, permettant de garantir un fonctionnement transparent des VNRs hébergées dans l'infrastructure, en cas de présence de pannes. Étant donné que la plupart de ces approches peuvent tout de même être affectées par un équilibrage insuffisant de la charge, dû à une sous-utilisation de certains liens de communication, (Khan *et al.*, 2016) ont alors proposé une amélioration dans la résolution du problème. Les auteurs ont développé une stratégie d'intégration des liens virtuels qui divise le trafic sur plusieurs chemins physiques, dans le but d'exploiter le potentiel des techniques de fractionnement de chemins pour assurer la survie de l'équipement réseau, tout en minimisant la redondance des ressources et le délai de fractionnement des chemins.

2.3.2 Minimisation de l'énergie

Certains travaux relatifs à la gestion du réseau intraCloud se sont intéressés à la dimension "vert" de l'infrastructure, en proposant des modèles de VNE permettant d'optimiser l'empreinte écologique du Cloud. En effet, l'exploitation à grande échelle des DCs et leur entretien nécessitent une énergie électrique considérable, se traduisant en retour par des émissions de carbone davantage importantes. En ce sens, plusieurs cadres d'assignation de charges dans le réseau Cloud visent en particulier à réduire l'empreinte carbone des DCs. (Justafort *et al.*, 2015) ont proposé un cadre de placement de VMs au sein d'un réseau distribué de plusieurs DCs, permettant notamment de minimiser l'impact écologique, en considérant divers aspects liés à la puissance des équipements de calcul, à leur consommation énergétique et à celle des ressources réseau. Les auteurs ont également proposé une extension dans leur travail (Justafort *et al.*, 2016), afin de considérer des applications plus complexes faisant intervenir du

trafic entre les VMs. (Khoshkholghi *et al.*, 2017), quant à eux, ont additionnellement considéré une dimension dynamique du processus de placement des VMs, en utilisant des techniques de migration de ces dernières entre les différentes entités hôtes, afin d'améliorer davantage l'utilisation des ressources informatiques et de réduire la consommation de l'énergie.

2.3.3 Maximisation des revenus/profits

Dans un tel contexte où les enjeux économiques demeurent importants pour le fournisseur du réseau Cloud, certains travaux se sont alors naturellement intéressés à définir des modèles maximisant les revenus ou profits du CP. L'optimisation de ces critères économiques se traduit notamment par une gestion adéquate de l'utilisation et de l'allocation des ressources de l'infrastructure Cloud, mais également celle des modèles de tarification et de facturation à soumettre aux locataires de services. Les auteurs (Chowdhury *et al.*, 2012) ont proposé des solutions de VNE maximisant les revenus du CP, tout en assurant une meilleure coordination entre l'intégration des VMs et celle des VLs, par opposition aux approches moins performantes, qui traitent ces entités séparément en deux phases séquentielles. Dans la même visée, (Hesselbach *et al.*, 2016) ont récemment proposé une stratégie de VNE basée sur l'algèbre des chemins, afin de maximiser le taux d'acceptation des demandes d'hébergement et les revenus du CP. (Zhang *et al.*, 2014) ont ajouté une étape supplémentaire à cette optimisation conjointe, en proposant une approche linéaire permettant de supporter la variation dynamique dans le temps, autant pour les ressources demandées que celles disponibles. Les auteurs ont alors mis en œuvre un mécanisme de re-allocation de ressources durant la durée de vie des requêtes, afin de continuer à assurer les services sans dégradation de la performance exigée. (Amokrane *et al.*, 2015) ont, quant à eux, développé un modèle considérant la variabilité dans le temps et dans l'espace de l'énergie renouvelable et des prix de celle-ci. Le modèle vise précisément à maximiser les profits du CP, ce qui se traduit dans leur approche par une maximisation des revenus et une minimisation des coûts opérationnels sur l'infrastructure, des coûts de migration des VMs et des pénalités liées aux émissions de carbone.

2.3.4 Optimisation multicritère

D'autres travaux de la littérature ont adopté une approche de résolution multiobjectif du problème du VNE intraCloud, dans le but d'offrir au CP la possibilité de privilégier, dans le même modèle, certains objectifs par rapport à d'autres selon ses priorités, mais également de lui permettre de faire des compromis entre des critères conflictuels. (Larumbe et Sanso, 2013), par exemple, ont proposé un modèle multiobjectif dans un cadre de VNE au sein d'un réseau constitué de plusieurs DCs. Leur approche permet de conjointement minimiser

les coûts d'exploitation des ressources serveur pour l'hébergement des VMs et des ressources réseau pour le routage du trafic, mais également les coûts de consommation d'énergie, les pénalités relatives à l'émission de carbone et au délai de routage, ainsi que les coûts d'opération et d'entretien des DCs. (Houidi *et al.*, 2015) ont également proposé un modèle multiobjectif prenant en compte plusieurs critères et contraintes de performance relatifs à la puissance de consommation en énergie de l'équipement physique, à la disponibilité des ressources et à l'équilibrage des charges. De plus, le modèle intègre des mécanismes de re-allocation de ressources, afin de considérer des cas de pannes de l'équipement et les variations dynamiques en demande de ressources des VNRs. (Melo *et al.*, 2013), quant à eux, se sont intéressés à développer un modèle de VNE en ligne, permettant d'optimiser conjointement l'utilisation des ressources et l'équilibrage des charges, ainsi que le délai de routage entre les DCs. Une extension de leur travail est réalisée par la suite (Melo *et al.*, 2015), dans le but de considérer également l'optimisation de la consommation de énergie de l'infrastructure. (Pyoung et Baek, 2018) ont également récemment proposé un cadre de VNE optimisant conjointement l'équilibrage des charges et les économies en énergie, afin de maximiser les profits générés en conséquence par le CP.

2.4 Approches de résolution du VNE

Les méthodes utilisées dans la littérature pour résoudre le problème combinatoire du VNE dans un domaine multiCloud, peuvent se classer en deux groupes, à savoir les méthodes exactes et les méthodes approchées.

2.4.1 Approches exactes

Les approches exactes permettent de trouver et de garantir une solution unique et optimale à un problème d'optimisation. Elles sont employées dans la littérature selon plusieurs méthodes présentées ci-dessous.

2.4.1.1 Méthodes par énumération

Les méthodes exactes les plus simples consistent en une énumération et évaluation de toutes les solutions possibles au problème traité, avant d'en sélectionner la meilleure. Cependant, vu l'éventail des possibilités de configurations de solution, il devient impraticable de toutes les énumérer, en particulier avec de grandes instances du problème et un nombre élevé de contraintes à vérifier. Les algorithmes par séparation et évaluation ou *Branch-and-Bound* sont par contre des méthodes exactes permettant aussi d'énumérer les solutions possibles du

problème, mais cela de façon plus intelligente que la méthode d'énumération classique. En effet, en pratique les solutions potentiellement de bonne qualité sont explorées, en ignorant celles détectées de moins bonne qualité ou non faisables. La distinction entre les solutions est effectuée grâce à certains critères définis selon la nature du problème, dont le plus souvent étant la borne supérieure ou inférieure de la fonction objectif. Certains outils informatiques d'optimisation comme (CPLEX, 2018), communément appelés solveurs, mettent en œuvre des algorithmes de *Branch-and-Bound*, permettant ainsi de résoudre de manière optimale des petites instances du problème en un temps raisonnable.

2.4.1.2 Programmation par contraintes

La méthode de programmation par contraintes consiste à modéliser un problème par un ensemble de relations logiques, des contraintes, imposant des conditions sur l'instanciation possible d'un ensemble fini de variables définissant une solution au problème. Un solveur de contraintes détermine une solution par des déductions logiques, en éliminant progressivement les configurations non faisables jusqu'à instancier chacune des variables à une valeur satisfaisant simultanément toutes les contraintes du problème. (Zou *et al.*, 2015) ont employé des techniques algorithmiques basées sur la programmation par contraintes, afin d'optimiser, dans le cadre d'un déploiement de VMs dans un environnement à multiples Clouds publics, la sélection des hôtes adéquats permettant de minimiser la latence de communication entre les VMs. (Coullon *et al.*, 2017) ont également proposé un modèle flexible de placement de VMs dans un environnement de Cloud fédéré ou hybride, basé sur la programmation par contraintes, afin de minimiser l'énergie totale utilisée.

2.4.1.3 Programmation dynamique

La programmation dynamique est une méthode exacte d'optimisation de processus de décisions séquentielles permettant de résoudre un problème complexe, en le décomposant en un ensemble de sous-problèmes plus simples. Une solution optimale est ensuite déterminée à partir de la résolution de chacun de ces sous-problèmes, en stockant leurs solutions à l'aide d'une structure de données basée sur une mémoire indexable permettant faciliter la recherche. Ainsi, la prochaine fois que le même sous-problème se produit, au lieu de recalculer sa solution, la méthode se contente alors de consulter la solution précédemment calculée, ce qui permet de gagner du temps de calcul. (Wang *et al.*, 2013) ont proposé une stratégie de réservation automatique d'instances de services basée sur la programmation dynamique, qui permet au VNP d'exploiter de manière optimale les avantages de tarification sur la réservation de ressources à long terme, dans le but de minimiser le coût d'approvisionnement des services. (Ayoubi *et al.*,

2015) ont également utilisé une approche de programmation dynamique dans leur modèle de VNE, afin de traiter des instances de réseaux virtuels à multidiffusion pouvant être résolues en temps polynomial.

2.4.1.4 Programmation mathématique

La programmation mathématique permet de modéliser un problème d'optimisation combinatoire sous la forme d'un système d'équations mathématiques, représentant des contraintes du problème, dans le but d'optimiser une fonction objectif donnée. Le modèle mathématique peut être linéaire ou non, dépendamment de la linéarité de la fonction objectif et des équations mathématiques utilisées pour exprimer les contraintes. Les variables du modèle mathématique peuvent être continues, discrètes (ILP), ou mixtes (MILP), et peuvent également être contraintes à prendre des valeurs binaires. Dans la littérature, nombreuses sont des solutions obtenues de manière optimale en formulant le problème du VNE multiCloud ou intraCloud au moyen de la programmation mathématique, plus précisément avec un modèle ILP (Leivadeas *et al.*, 2013; Dietrich *et al.*, 2015; Mechtri *et al.*, 2017; Li *et al.*, 2016) ou MILP (Larumbe et Sanso, 2013; Houidi *et al.*, 2015). La programmation linéaire est également souvent utilisée avec d'autres méthodes, telles que la programmation par contraintes, dans le but d'améliorer davantage l'efficacité des algorithmes proposés. L'approche d'approvisionnement de VMs proposée par (Chaisiri *et al.*, 2012) et (Ran *et al.*, 2016), qui inclut une formulation mathématique déterministe équivalente du problème, utilise un modèle entier de programmation stochastique résolu à plusieurs niveaux de l'optimisation du processus.

Les approches exactes, même si elles peuvent garantir l'obtention de solutions optimales au problème du VNE, introduisent toutefois des temps de résolution augmentant de manière exponentielle lorsque la taille des instances du problème devient grande. De ce fait, des approches approximatives sont souvent privilégiées au détriment de la résolution exacte, particulièrement dans le cas des problèmes NP-difficiles tels que le VNE.

2.4.2 Approches approximatives

Les méthodes approximatives représentent des algorithmes de résolution qui permettent, par une approche intuitive et heuristique se basant sur l'interprétation et l'exploitation de la structure d'un problème combinatoire, de trouver de bonnes solutions en un temps raisonnable. Les approches approximatives utilisées généralement dans la littérature pour résoudre le problème du VNE multiCloud ou intraCloud, regroupent, entre autres, les heuristiques de construction progressive, les métaheuristiques, les méthodes hybrides et les méthodes d'apprentissage automatique.

2.4.2.1 Heuristiques constructives

Les heuristiques constructives sont généralement les approches approximatives les plus rapides et les plus simples à implémenter. Elles permettent de construire progressivement une solution initialement vide, en y ajoutant au fur et à mesure des composants de solution définis de manière appropriée, en se basant sur des règles de décision prédéfinies. Ceci est fait jusqu'à ce que la solution soit complète ou que d'autres critères d'arrêt soient satisfaits. Toutefois, durant les étapes de construction de la solution, aucune prise de décision antérieure n'est remise en cause ou modifiée. Les heuristiques de construction progressive utilisées dans la littérature en lien avec le VNE incluent, entre autres, les méthodes gloutonnes. Ces dernières consistent généralement à assigner chacune des VMs au meilleur CP, DC ou serveur, et par la suite à assigner chacune des VLs au meilleur chemin entre ces éléments hôtes des VMs. De ce fait, les algorithmes gloutons, même s'ils peuvent assurer une optimalité locale pour une VM ou VL donnée, ne peuvent pas garantir une bonne ou même réalisable solution, considérant toutes les interactions entre les entités du problème et les contraintes liées. Tout de même, certains auteurs utilisent une approche de résolution du problème basée sur les méthodes gloutonnes, en raison de leur faible complexité d'implémentation et de leur rapidité à trouver une solution. Par exemple, (Bellur *et al.*, 2014) ont proposé un algorithme glouton d'assignation de VMs dans un environnement multiCloud, en classant au préalable les sites Cloud par ordre croissant selon le coût d'approvisionnement et en plaçant les VMs une à une de manière gloutonne, dans le but de minimiser le coût total de la demande. (Nia *et al.*, 2017) ont également employé une méthode gloutonne pour résoudre leur modèle mathématique de VNE dans un environnement de DCs distribués, en définissant d'abord un critère de mesure permettant de classer les VMs et de les assigner par ordre aux DCs, pour utiliser par la suite la méthode d'algèbre des chemins pour router les VLs.

Du fait de leur particularité en rapidité de résolution, les méthodes gloutonnes sont en outre souvent utilisées pour déterminer une configuration initiale de solution, pouvant par la suite être améliorée par d'autres méthodes, telles que la recherche locale ou les métaheuristiques.

2.4.2.2 Métaheuristiques

Les métaheuristiques (Blum et Roli, 2003; Boussaïd *et al.*, 2013) sont des stratégies qui permettent de guider le processus de recherche de solutions, en utilisant des techniques aléatoires et itérativess pouvant aller d'une simple procédure de recherche à un processus d'apprentissage complexe. Le but est notamment d'explorer et d'exploiter efficacement l'espace de recherche, afin de trouver des solutions optimales dans le meilleur des cas, ou alors sous-optimales. Les métaheuristiques ne sont pas spécifiques à un problème combinatoire

donné. En effet, elles sont souvent adaptées à la résolution d'un problème en particulier, en définissant des mécanismes allant d'une approche simple de représentation des solutions au problème, à des techniques plus approfondies d'exploration de l'espace de recherche et de paramétrage des algorithmes. Les métaheuristiques sont généralement non déterministes et ne garantissent aucune optimalité des solutions obtenues, mais elles peuvent néanmoins incorporer des mécanismes basés sur l'expérience accumulée durant la recherche, pour permettre de mieux guider la suite de la procédure et éviter des pièges dans des zones confinées de l'espace de recherche.

Parmi les métaheuristiques, on distingue en particulier les méthodes à solution unique basée sur la recherche locale et les méthodes à base de population.

Méthodes de recherche locale : Les algorithmes de recherche locale ou Local Search (LS) partent d'une solution initiale, souvent aléatoire ou gloutonne, et tentent de manière itérative de remplacer la solution actuelle par une solution meilleure dans un voisinage défini de manière appropriée de la solution actuelle, en appliquant des mouvements adaptés au problème à traiter. Ainsi, l'espace de recherche est exploré progressivement en passant d'une solution à l'une de ses voisines, et la recherche peut s'arrêter selon plusieurs critères d'arrêt, tels que le nombre d'itérations total à faire, le nombre d'itérations sans modification de la meilleure solution, le temps limite imposé ou une distance moyenne acceptable par rapport à une borne.

La méthode de recherche locale la plus simple est l'algorithme de descente, qui part d'un point initial et qui, à chaque itération, choisit la meilleure solution dans le voisinage de la solution actuelle, et reprend la descente à partir de cette nouvelle solution. La recherche s'arrête au moment où aucune amélioration de la meilleure solution actuelle n'est observée entre deux itérations successives. (Zeng *et al.*, 2014) ont proposé une stratégie centralisée basée sur l'algorithme de descente, afin de gérer le routage de trafics sensibles au délai, entre plusieurs DCs dans un réseau intraCloud distribué. Bien que rapide, le principal défaut de l'algorithme de descente est son arrêt prématuré au premier optimum local rencontré, qui peut de surcroît être très loin de l'optimum global.

D'autres approches de LS peuvent être utilisées pour pallier ce problème. L'une d'être elles est la Recherche Locale Itérée ou Iterated Local Search (ILS), qui consiste à appliquer une perturbation contrôlée sur la solution relative à l'optimum local, afin de reprendre la descente à partir d'une nouvelle configuration de la solution. Les perturbations sont souvent répétées pendant un certain nombre d'itérations défini, dans le but de mieux explorer l'espace de recherche. (Leivadeas *et al.*, 2013) ont proposé une approche de résolution de leur modèle

de partitionnement multiCloud basée sur ILS, afin de pouvoir traiter de grandes instances de VNRs. Les auteurs (Justafort V. D., 2015) ont également proposé une adaptation d'ILS pour résoudre leur modèle de placement de VMs à l'échelle de plusieurs DCs, dans le but de minimiser l'empreinte carbone.

La recherche à voisinage variable ou Variable Neighborhood Search (VNS) est une méthode de LS qui se base sur un changement de type de voisinage au courant de la recherche locale. L'algorithme part d'une solution initiale avant d'entamer la descente. Si un optimum local est rencontré, une procédure prédéterminée de changement de méthode de voisinage est alors appliquée sur la solution courante. L'algorithme utilise donc plusieurs méthodes pour déterminer le voisinage d'une solution donnée, afin de pouvoir, à des moments de la recherche, s'échapper d'un optimum local et trouver de meilleures solutions ailleurs.

La technique du Récuit Simulé ou Simulated Annealing (SA), dérivée du domaine de la physique, est également utilisée pour améliorer l'algorithme de descente. Le SA commence par une solution initiale associée d'une certaine température de base et, à chaque itération, une configuration voisine de la solution actuelle est choisie au hasard. Cette solution voisine est automatiquement acceptée si la température qui lui est associée est meilleure que celle de la solution actuelle. Dans le cas contraire, elle n'est acceptée comme nouvelle solution qu'avec une certaine probabilité. Si la solution voisine est refusée, une nouvelle solution est tout de même choisie au hasard dans le voisinage de la solution actuelle, et ainsi de suite jusqu'à atteindre la condition d'arrêt de l'algorithme. (Addya *et al.*, 2017) ont utilisé la méthode du SA pour la résolution de leur stratégie de placement de VMs dans une infrastructure Cloud, visant à minimiser la puissance de consommation des VMs tout en maximisant les revenus du CP.

L'algorithme de Recherche Taboue ou Tabu Search (TS) (Glover, 1989, 1990) est très populaire en matière de LS, car permettant d'éviter efficacement les optimums locaux et les cycles dans le parcours de l'espace, grâce notamment à l'utilisation de mécanismes de mémoire flexible. Partant d'une solution initiale S_0, l'algorithme de TS choisit toujours, à chaque itération i, la meilleure solution S_{i+1} dans le voisinage de la solution actuelle S_i, même si elle peut dégrader le coût de la solution S_i. Cependant, il n'est pas impossible que la solution courante S_i soit également la meilleure solution dans le voisinage de S_{i+1}. Pour éviter de cycler autour de ces deux solutions, l'algorithme utilise un mécanisme de mémoire à court terme, faisant référence à une liste taboue, qui permet de conserver une historique d'attributs relatifs aux récentes solutions déjà visitées, afin d'interdire de choisir une solution répondant aux critères qui figurent dans la liste pendant un certain nombre d'itérations. Toutefois, le statut tabou d'une solution peut être levé si cette dernière répond à un critère d'aspiration,

lequel peut, par exemple, permettre son acceptation si la solution est meilleure que toutes les solutions obtenues jusque-là. D'autres mécanismes de mémoire à moyen terme (Intensification) et à long terme (Diversification) peuvent également être employés. Plus précisément, l'Intensification permet d'explorer plus profondément les alentours des meilleures solutions trouvées depuis le début de la recherche, alors que la Diversification permet de relancer la recherche à partir de régions inexplorées, dans le but d'exploiter davantage l'espace des solutions possibles. (Larumbe et Sanso, 2013) ont proposé une adaptation de TS comme approche de résolution de leur modèle multiobjectif de placement de VMs et de VLs dans un réseau distribué de DCs. (Ayoubi *et al.*, 2016) ont également adopté une méthode de résolution basée sur TS, dans le cadre de leur modèle dynamique et de fiabilité de VNE.

Méthodes à base de population : Contrairement aux méthodes de recherche locale qui font évoluer itérativement une solution courante unique, les méthodes à base de population travaillent, quant à elles, simultanément sur un ensemble de solutions. Un processus cyclique, composé de plusieurs itérations d'une étape de coopération et d'une étape subséquente d'adaptation individuelle, est alors mis en œuvre. À l'étape de coopération, les solutions d'une population courante sont comparées entre elles suivant des critères bien définis, puis combinées afin de produire de nouvelles solutions héritant des meilleurs attributs de chaque solution de la population. À l'étape d'adaptation individuelle, chaque nouvelle solution peut évoluer de manière indépendante, et ainsi de suite.Le processus peut s'arrêter selon les mêmes conditions d'arrêt que celles considérées avec les approches de LS, ou alors l'arrêt peut se produire si les solutions de la population sont jugées trop similaires.

L'algorithme génétique ou Genetic Algorithm (GA) est une méthode évolutionnaire à base de population très populaire, dont le principe est basé sur la sélection naturelle et la génétique. Le GA part d'une population initiale composée d'un ensemble fini d'individus (chromosomes). Par la suite, à chaque itération (génération), une succession d'opérations évolutionnaires de sélection, de croisement, de mutation et d'évaluation, est appliquée aux individus de la population actuelle, afin de produire la génération suivante de population. Durant ces opérations, certains individus disparaissent, d'autres se reproduisent, et les meilleurs survivent au fil des générations. Une fois le critère d'arrêt atteint, le meilleur individu survivant est retenu comme la meilleure solution finale. (Pathak et Vidyarthi, 2017; Zhang *et al.*, 2017) ont appliqué une adaptation du GA dans leur modèle de VNE, dans le but de maximiser le taux d'acceptation des VNRs, et ainsi les revenus des CPs. Bien que efficace, le GA peut toute fois être très couteux en temps de résolution.

L'optimisation par essaim de particules ou Particle Swarm Optimization (PSO), s'inspirant des comportements collectifs d'animaux en déplacement, est un autre algorithme à base de

population également utilisé dans la résolution du problème du VNE. L'algorithme PSO travaille sur une population, appelée essaim, constituée d'un ensemble d'individus (particules), représentant chacun une solution potentielle au problème. Chaque particule part d'une position généralement aléatoire dans l'espace de recherche. À chaque itération, chacune des particules se déplace dans l'espace avec une vitesse adaptable, et conserve dans sa propre mémoire la meilleure position jamais atteinte et celle dans son voisinage. Ces informations vont par la suite permettre à chaque particule de déterminer la prochaine direction qu'elle doit prendre et à quelle vitesse de déplacement elle doit évoluer. À la fin de l'algorithme, la particule ayant la meilleure position dans l'espace de recherche est retenue comme la meilleure solution. (Zhang *et al.*, 2016b) ont proposé une adaptation de l'algorithme de PSO, afin de résoudre un modèle multiobjectif de VNE permettant de maximiser les revenus du CP, tout en minimisant les coûts de consommation en énergie au sein du réseau.

L'optimisation par colonies de fourmis ou Ant Colony Optimization (ACO) (Dorigo et Stützle, 2010) est une métaheuristique à base de population inspirée du comportement naturel des colonies de fourmis. Un ensemble fini de fourmis artificielles explorent parallèlement et de manière itérative l'espace de recherche. À chaque itération, chaque fourmi construit progressivement une solution initialement vide, en y ajoutant au fur et à mesure des composants. Chacun de ces derniers est choisi selon une décision locale stochastique, basée sur des informations heuristiques et des pistes de phéromones artificielles associées aux différents composants potentiels. Une fois que toutes les fourmis aient construit une solution complète, chacune d'elles (ou la meilleure d'entre elles) dépose une quantité de phéromone artificielle sur les composants choisis par elle (ou uniquement sur ceux choisis par la meilleure fourmis), lors de la construction de la solution. Ceci permet aux fourmis de se guider entre elles vers les meilleures pistes dans l'espace, et de perfectionner en permanence les solutions à obtenir dans les recherches futures. Outre l'activité des fourmis, l'algorithme ACO inclut deux procédures supplémentaires, à savoir l'évaporation des pistes de phéromone, qui aide en l'occurrence les fourmis à découvrir de nouvelles trajectoires dans l'environnement et de ne pas être piégées dans des optima locaux, et un mécanisme optionnel de renforcement, qui peut être mis en œuvre par des actions centralisées ne pouvant être effectuées par une seule fourmi, telles que l'ajout supplémentaire d'une quantité de phéromone sur une piste prometteuse ou l'activation d'une procédure d'optimisation locale. Certains auteurs, tels que (Guan *et al.*, 2015; Zhu et Wang, 2016; Wang *et al.*, 2017b; Zong *et al.*, 2018), ont utilisé des adaptations variées de l'algorithme d'ACO pour résoudre le problème de la VNE, profitant ainsi des larges capacités d'adaptation de l'algorithme. Cependant, la recherche de solutions avec l'ACO de base peut souvent tomber dans des optima locaux (Dorigo et Stützle, 2010). Par conséquent, différentes techniques de renforcement de l'algorithme, telles qu'une hybridation avec une

méthode de LS, peuvent aider à mieux explorer l'espace de recherche, afin d'aboutir à une meilleure qualité de solutions.

2.4.2.3 Méthodes hybrides

Au cours des dernières années, une technique portant sur le principe de l'hybridation (Blum *et al.*, 2011) a beaucoup retenu l'attention des communautés de chercheurs, consistant généralement à combiner deux méthodes de résolution différentes, afin d'exploiter le potentiel complémentaire de chacune d'elles. L'hybridation peut faire intervenir, par exemple, une approche exacte de programmation mathématique et une approche heuristique, ou alors deux approches heuristiques. Les techniques d'hybridation peuvent utiliser une combinaison intégrative de deux méthodes, dont l'une faisant partie intégrante de l'autre, ou une combinaison collaborative, dans laquelle les deux méthodes sont exécutées séparément (de manière séquentielle, imbriquée ou parallèle), mais les résultats de l'une des méthodes sont utilisés par l'autre. Certains travaux ont démontré les avantages d'utiliser des métaheuristiques hybrides pour résoudre le problème du VNE, en combinant principalement des algorithmes gloutons ou basés sur la population, avec un opérateur de recherche locale. (Inführ et Raidl, 2016) ont proposé une approche mémétique pour résoudre leur modèle de VNE, en combinant le GA et l'algorithme de VNS, utilisé comme technique d'intensification. Les auteurs (Wang *et al.*, 2017a) ont utilisé une approche hybride de résolution combinant les algorithmes de PSO, de TS et de SA, afin d'améliorer le taux d'acceptation et les revenus du CP au sein d'un intraCloud. (Araújo *et al.*, 2018) ont proposé une stratégie de VNE basée sur l'algorithme de Greedy Randomized Adaptative Search Procedure (GRASP) combiné à la méthode de VNS, afin de garantir des délais minimaux de routage, une réduction énergétique et un équilibrage de charges dans un réseau multidomaine. Les auteurs (Justafort *et al.*, 2016) ont également utilisé une approche de résolution hybride d'Iterated Tabu Search (ITS), combinant les algorithmes d'ILS et de TS. Les résultats ont pu d'ailleurs montrer que l'algorithme d'ITS donne de meilleures performances, comparé aux deux méthodes employées séparément.

2.4.2.4 Méthodes d'apprentissage automatique

Au cours des dernières années, l'apprentissage automatique a permis de réaliser des avancées remarquables, en offrant des techniques de pointe permettant de traiter des tâches complexes pour des problèmes de prise de décision. En effet, contrairement aux métaheuristiques qui doivent appliquer le modèle défini et le résoudre pour chaque requête qui arrive dans le système, les algorithmes d'apprentissage automatique, quant à eux, estiment un modèle en traitant d'abord une grande quantité de données collectées au cours d'une période, puis se

basent sur ce modèle pour estimer ou prédire instantanément le résultat pour de nouvelles données entrant dans le système.

Les algorithmes basés sur l'apprentissage automatique sont récemment proposés pour traiter des aspects dynamiques et adaptatifs dans l'allocation de ressources. (Mijumbi *et al.*, 2014) ont appliqué un agent d'apprentissage par renforcement basé sur l'algorithme de Q-learning, afin de mettre en place un système de gestion de ressources décentralisé, dont le rôle est d'augmenter ou de réduire les ressources allouées à une certaine VNR déjà intégrée dans le domaine sous-jacent. (He *et al.*, 2018) utilisent également des techniques d'apprentissage automatiques, afin de proposer une approche de VNE visant à optimiser dynamiquement plusieurs objectifs conflictuels faisant référence à la maximisation du taux d'acceptation et à la réduction de la consommation en énergie. Les auteurs (Yao *et al.*, 2018) ont introduit, quant à eux, une méthode d'apprentissage par le renforcement dans un cadre de placement de nœuds virtuels dans un réseau physique, afin d'optimiser l'utilisation des ressources dans le réseau du fournisseur.

2.5 Analyse des travaux présentés dans la littérature

L'analyse de la littérature actuelle nous permet d'observer que cette dernière est assez peu fournie en ce qui concerne le problème du VNE dans un contexte multiCloud. En effet, nombreux sont les efforts sur le sujet du VNE qui ont été apportés dans un cadre ne faisant intervenir qu'un seul CP. Les travaux proposés se sont alors orientés sur plusieurs modèles de VNE intraCloud, dans le but d'optimiser l'utilisation des ressources et l'équilibrage des charges (Ayoubi *et al.*, 2015, 2016; Shahriar *et al.*, 2017; Khan *et al.*, 2016), ou de minimiser l'énergie et l'impact écologique (Justafort *et al.*, 2015; Khoshkholghi *et al.*, 2017), ou alors de maximiser les revenus du CP (Chowdhury *et al.*, 2012; Hesselbach *et al.*, 2016; Zhang *et al.*, 2014; Amokrane *et al.*, 2015). Les travaux les plus intéressants se basent tout de même sur des modèles multiobjectif de représentation du problème d'hébergement intraCloud des demandes (Houidi *et al.*, 2015; Larumbe et Sanso, 2013; Melo *et al.*, 2015; Pyoung et Baek, 2018). De telles approches permettent notamment de refléter plus adéquatement la réalité, en offrant au CP la possibilité de considérer simultanément plusieurs aspects critiques à optimiser dans son réseau, tout en lui laissant la flexibilité quant au choix de leur priorité dans la même fonction multicritère. Cependant, tel que mentionné précédemment, les solutions développées dans la littérature pour résoudre le problème du VNE intraCloud se basent essentiellement sur une maîtrise complète et détaillée du réseau d'infrastructure sous-jacent, et sont généralement orientées dans les intérêts du CP. De ce fait, ces solutions ne peuvent être appliquées dans le cadre d'un réseau multiCloud, qui nécessite d'une part une phase préalable de sélection

des différents CPs selon des politiques restrictives de confidentialité limitant les informations sur le réseau global. D'autre part, un tel contexte fait intervenir différents acteurs ayant des objectifs d'optimisation conflictuels. Toutefois, tel qu'observé dans la littérature, ces solutions sont souvent utilisées pour résoudre la phase de VNE intraCloud, subséquemment à la phase de sélections de CPs. Cette démarche permet souvent d'évaluer et de valider la stratégie de partitionnement des VNRs adoptée.

À l'égard du problème émergent du VNE dans un environnement multiCloud, les quelques travaux recensés dans la littérature, portant intérêt aux objectifs du SP, se sont majoritairement intéressés à la minimisation des coûts d'approvisionnement pour ce dernier (Dietrich *et al.*, 2015; Gong *et al.*, 2016; Mano *et al.*, 2016; Chaisiri *et al.*, 2012; Ran *et al.*, 2016). Cependant, dans la plupart des cas, les modèles proposés ne considèrent que des VMs à traiter, sans prendre en compte le trafic pouvant exister entre elles (Chaisiri *et al.*, 2012; Ran *et al.*, 2016). D'autres ne considèrent que des coûts de ressources fixes (Dietrich *et al.*, 2015; Gong *et al.*, 2016; Mano *et al.*, 2016), et n'intègrent donc pas des mécanismes de tarification dynamique basée sur la disponibilité des ressources chez les CPs ou sur la durée de vie des requêtes à héberger. De ce fait, à chaque arrivée de requêtes à traiter dans le système, ces modèles ont toujours tendance à sélectionner les mêmes CPs ayant les prix les plus réduits, ce qui ne permet pas de garantir un taux d'acceptation élevé et un degré de performance satisfaisant toutes les contraintes de QdS imposées par le SP.

Les autres travaux qui visent par contre la maximisation du taux d'acceptation et de la performance des VNRs partitionnées, sans tenir compte des aspects économiques, considèrent souvent certaines informations normalement dissimulées par les CPs pour réaliser leurs modèles de partitionnement (Leivadeas *et al.*, 2013), ce qui ne permet pas d'apprécier l'efficacité de ces derniers vis-à-vis des politiques de confidentialité des CPs à respecter. D'autres travaux prenant en compte les restrictions sur le domaine (Li *et al.*, 2016; Mechtri *et al.*, 2017), manquent de considérer dans leurs modèles de partitionnement toutes les ressources et contraintes conjointement relatives aux VMs et aux VLs, ce qui ne permet pas de représenter des applications complexes avec tous les besoins associés en performance et en QdS.

Par ailleurs, ces travaux proposés dans le cadre du VNE multiCloud, s'intéressant soit uniquement à la minimisation des coûts d'approvisionnement pour le SP, ou soit à l'optimisation des performances et de la QdS des VNRs, ne laissent donc pas l'opportunité au SP de sélectionner les différents services et ressources Cloud selon le niveau de performance et de QdS souhaité, tout en minimisant, en même temps, les coûts d'approvisionnement y résultant.

Quant aux approches de résolution du problème recensées dans la littérature, il est manifeste que les méthodes exactes peuvent certes procurer des solutions optimales, cependant,

elles ne sont limitées que sur des instances de petite taille capables d'être résolues en un temps raisonnable. La résolution d'une telle complexité de problème combinatoire nécessite indubitablement le développement de méthodes approximatives rapides et performantes, permettant de fournir des solutions acceptables en un temps de calcul polynomiale, notamment pour des instances de grande taille du problème. Bien que plusieurs approches approximatives de résolution aient été proposées dans la littérature, les métaheuristiques ou méthodes hybrides semblent les plus adéquates pour traiter le problème. En effet, les méthodes d'apprentissage automatique, bien que intéressantes et sophistiquées, nécessitent toutefois une base de données étiquetées énorme et temporellement couteuse pour entraîner et tester le système de VNE, ce qui peut être laborieux à acquérir au final.

En regard de l'état actuel de la littérature sur le sujet émergent du VNE dans un environnement multiCloud, et en particulier pour ce qui a trait au problème du partitionnement des VNRs, nous nous efforcerons dans cette thèse à intégrer tous les aspects précédemment mentionnés. Le cadre de VNE multiCloud que nous proposons dans ce travail permettra de considérer des applications allant des plus simples aux plus complexes, et prendra en compte l'ensemble des besoins en performance et contraintes de QdS associés aux demandes, ainsi que les politiques restrictives des CPs. Le but sera, plus précisément, de conjointement maximiser le taux d'acceptation des demandes à héberger dans les différents Clouds et d'optimiser les performances des applications, tout en minimisant, au mieux possible pour le SP, les coûts d'approvisionnement des ressources et services. Le cadre proposé intégrera également des approches de résolution automatiques et avancées, permettant de pallier la nature NP-Difficile du problème.

CHAPITRE 3 DÉMARCHE MÉTHODOLOGIQUE DE L'ENSEMBLE DU TRAVAIL DE RECHERCHE

Cette thèse vise à concevoir un cadre hiérarchique d'intégration des requêtes de réseaux virtuels (VNRs) dans un environnement multiCloud, permettant d'optimiser, à moindre coût d'approvisionnement pour le SP, la performance et la QdS de ses applications qu'il héberge chez les CPs. Le cadre à concevoir inclut notamment l'élaboration d'une stratégie de partitionnement des VNRs parmi les multiples CPs, et l'adoption d'une méthode subséquente d'hébergement des segments de VNRs résultants, dans le réseau intraCloud de chaque CP choisi. Dans ce chapitre, nous décrivons toute la démarche méthodologique mise en œuvre pour répondre à la visée de cette thèse, et nous mettons notamment en évidence les liens existant entre les différents objectifs énoncés à la Section 1.3 et les articles scientifiques présentés dans les prochains chapitres du manuscrit. De manière générale, les modèles développés dans ce cadre de VNE multiCloud, sont évalués sur un ensemble de critères de performance définis dans la suite, et les travaux réalisés sont organisés en trois (3) grands volets.

3.1 Définition des critères de performance

Les critères de performance et d'économie utilisés dans nos travaux pour évaluer l'efficacité de la stratégie de partitionnement proposé dans le cadre du VNE multiCloud, sont résumés sur la Figure 3.1 et décrits ci-dessous :

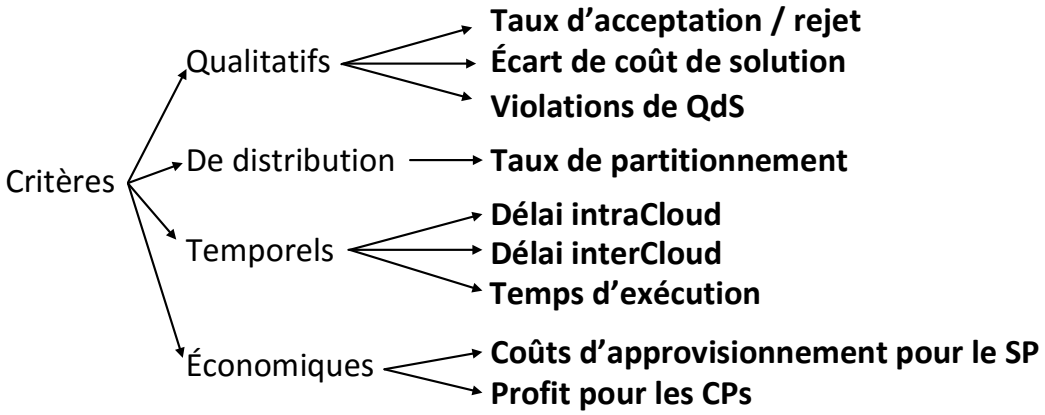

Figure 3.1 Critères de performance et d'économie

— **Taux d'acceptation**, qui représente une des mesures les plus concluantes pour évaluer l'efficacité d'une stratégie de partitionnement de VNRs basée sur la performance des applications à héberger. Le taux d'acceptation représente le quotient du nombre de VNRs hébergées avec succès par les CPs, y compris les connexions interCloud réussies, sur le nombre total de VNRs soumises.

— **Écart de coût de solution**, traduisant la distance moyenne en pourcentage entre le coût d'une solution obtenue avec une méthode approximative et celui obtenu avec la solution optimale ou bornée. Plus cette distance est petite, plus elle traduit la qualité de la solution générée par l'approche heuristique.

— **Violations de délai**, qui résultent d'une intégration de VNR qui ne satisfait pas les contraintes de délai maximal établies par le SP sur les communications inter-VMs. Ce critère est mesuré simplement comme étant le nombre de VLs dont le délai de routage dépasse le délai maximal autorisé.

— **Taux de partitionnement**, qui est défini comme étant le quotient du nombre de segments de requêtes assignés aux CPs sélectionnés à chaque VNR soumise, divisé par le nombre total de CPs. Dans le meilleur des cas, une stratégie efficace devrait générer un faible taux de partitionnement, tout en procurant un taux d'acceptation élevé. En effet, un taux de partitionnement des VNRs inutilement élevé devrait être évité, car cela introduit généralement un grand nombre de VLs transitant par les chemins interCloud, ce qui peut entraîner des délais de communication élevés.

— **Délai**, qui représente l'une des métriques de performance qui influencent le plus la QdS. Le délai pour un paquet transitant entre deux VMs communicantes, est défini comme le délai total sur le chemin entre les deux DCs hôtes correspondants. Si les deux DCs appartiennent au même CP, le délai évalué représentera le délai intraCloud, sinon le délai interCloud.

— **Temps d'exécution**, représentant le temps de calcul nécessaire au solveur CPLEX et aux algorithmes heuristiques pour exécuter les modèles et générer une solution. Dans le cas des approches approximatives, l'objectif est généralement de trouver un excellent compromis entre écart de coût des solutions obtenues et temps d'exécution des algorithmes.

— **Coût total d'approvisionnement pour le SP**, défini à la Section 1.1.5.4 et qui inclut le coût d'approvisionnement des ressources serveur et réseau, le coût des services de transit et de communication interCloud, que le SP cherche à minimiser.

— **Profit pour le CP**, que ce dernier cherche généralement à maximiser et qui définit la différence entre les revenus du CP après hébergement d'une demande et les couts d'exploitation de son infrastructure.

3.2 Volet 1 : Stratégie de partitionnement multiCloud des VNRs basée sur un modèle de programmation mathématique

Dans le but d'optimiser la performance et la QdS des applications à héberger dans l'environnement multiCloud et de maximiser le taux d'acceptation des demandes, les deux (2) premiers objectifs de la thèse consistent à proposer une modélisation mathématique du problème du partitionnement des VNRs, et à concevoir un modèle de programmation mathématique permettant de résoudre de manière exacte la stratégie de partitionnement proposée. Par la suite, il est question d'adopter un modèle mathématique de résolution du problème d'hébergement intraCloud pour chacun des CPs sélectionnés par la stratégie de partitionnement, et d'évaluer l'efficacité de cette dernière en la comparant avec d'autres modèles de la littérature, selon plusieurs critères de performance et de QdS. Ces quatre (4) objectifs ont été atteints dans l'article intitulé « *A QoS-based splitting strategy for a resource embedding across multiple cloud providers* » (Diallo *et al.*, 2018a), présenté au chapitre 4, dans lequel une stratégie de partitionnement des VNRs basée sur la QdS est proposée, dans un cadre de VNE dans un environnement multiCloud. En raison de règles éditoriales limitant le nombre de pages dans les articles, le modèle MILP relatif au problème d'hébergement intraCloud des segments de VNRs est présenté dans l'annexe de l'article.

3.2.1 Modélisation mathématique du problème de partitionnement

La stratégie de partitionnement des VNRs proposée dans ce premier volet, permettant d'optimiser la performance et la QdS des VMs et du trafic inter-VMs, est modélisée par le biais d'équations mathématiques et formulée dans un modèle de programmation linéaire en nombres entiers (ILP). Les différentes étapes réalisées à cette fin sont décrites ci-dessous.

3.2.1.1 Composition du référentiel d'informations

Afin de fournir une meilleure efficacité de la stratégie de partitionnement proposée, nous avons analysé les informations auxquelles le VNP peut accéder pour procéder à la meilleure sélection des CPs hôtes, sans pour autant entraver leurs politiques de confidentialité. Sur la base des références connexes étudiées, pour chacun des CPs, les informations relatives aux capacités maximales disponibles pour un type d'instances de ressources informatiques et aux capacités minimales de bande passante garantie pour un type de liaison réseau donné, sont incorporées dans le référentiel d'informations. De plus, pour enrichir davantage la visibilité du VNP sur l'environnement multiCloud, certaines informations statistiques ou de garantie de performance relatives aux réseaux PoPs des CPs et des liaisons interCloud, ont également

pu être ajoutées au référentiel. Le VNP ne disposant que d'une connaissance très limitée et agrégée de l'environnement multiCloud, cette analyse de l'entrepôt de données était essentielle pour l'efficacité et la conformité de la stratégie de partitionnement à concevoir.

3.2.1.2 Modélisation de l'environnement multiCloud et des VNRs

Les réseaux multiCloud et VNRs sont représentés dans notre modèle par des graphes non orientés. La modélisation de l'environnement multiCloud est réalisée selon la vue topologique du réseau accessible par le VNP et les informations contenues dans le référentiel défini précédemment. L'ensemble des chemins existant entre les CPs sont également modélisés, incluant pour chacun le délai moyen de propagation sur chaque lien interCloud, le nombre de domaines PoPs que le chemin traverse, ainsi que le délai moyen pour transiter par ces derniers. Sur observation du modèle IaaS de (Amazon, 2017), les VMs sont classifiées par catégorie selon un ensemble d'attributs fonctionnels définis (type des VMs demandées, environnement de virtualisation, systèmes d'exploitation, localisation géographique, paramètres de QdS et de sécurité, etc.). Les VMs sont aussi accompagnées d'un ensemble de ressources informatiques (CPU, mémoire, disque, etc.), chacune étant associée d'une certaine quantité requise. De même, les VLs sont définies selon un type de liaison réseau, une quantité de bande passante nécessaire à satisfaire, ainsi qu'un délai de routage maximal à respecter. Les catégories de VMs et les types de VLs sont disponibles différemment d'un Cloud à un autre, et un seul CP n'offre pas toutes les catégories possibles de VMs. De même, dans le domaine interne d'un CP, tous les DCs ne supportent pas les mêmes catégories de VMs. De ce fait, une première étape de mise en correspondance entre les catégories de VMs offertes et celles formulées dans les VNRs est effectuée, avant de procéder à la stratégie de partitionnement selon les capacités agrégées disponibles chez les différents CPs.

3.2.1.3 Formulation mathématique de la stratégie de partitionnement et définition du modèle ILP proposé

Nous avons utilisé un modèle ILP pour formaliser la stratégie de partitionnement des VNRs proposée sous forme d'un problème de maximisation sous un ensemble de contraintes. Ces dernières incluent les contraintes imposées par le SP, ainsi que les contraintes d'intégrité liées au modèle et à celles de la capacité maximale des CPs. Chacun des termes intervenant dans la fonction objectif sont préalablement définis par des équations mathématiques exprimant la demande et l'offre dans l'environnement multiCloud. La disponibilité d'une ressource serveur ou réseau chez un CP est traduite sous forme de "quota" normalisé entre 0 et 1, et celle entre deux CPs est définie en fonction de leurs "quotas" respectifs et du nombre de PoPs intermé-

diaires. De plus, chacune des quantités de ressources demandées par une VM est pondérée entre 0 et 1, selon l'importance de chacune d'elle dans la demande, ce qui permet d'assigner efficacement les VMs en fonction du type d'application qu'elles supportent. Le modèle ILP défini permet donc de proposer des solutions de partitionnement de VNRs optimisant à la fois les performances et la QdS des VMs et des VLs, en sélectionnant de manière stratégique les CPs capables d'offrir les capacités de ressources serveur et réseau les plus élevées aux emplacements souhaités. Le modèle défini minimise également le nombre de sauts (en termes de réseaux de transit de PoP), ainsi que les délais de communication entre les CPs hôtes, tout en évitant la violation des délais maximaux spécifiés par le SP.

3.2.2 Adoption d'un modèle mathématique de résolution du problème d'hébergement intraCloud

Après avoir sélectionné stratégiquement les CPs capables de répondre aux exigences de la demande, et partitionné efficacement les VNRs, il nous vient à présent de résoudre le problème d'hébergement des segments de VNRs dans les différentes infrastructures intraCloud choisie. Pour cela, pour chacun des CPs qui reçoit une VNR partielle, nous adoptons une solution multicritère sur la base des principes énoncés par (Larumbe et Sanso, 2013). L'approche multiobjectif adoptée permet à chaque CP, dans le meilleur des cas, d'intégrer le segment de VNR qui lui est attribué dans son Cloud, en utilisant un modèle de programmation MILP. Ce dernier vise à minimiser le coût total d'hébergement de la demande, incluant le coût d'exploitation des ressources serveur et réseau, les coûts énergétiques et environnementaux, tout en offrant la meilleure QdS en réduisant principalement le délai de routage global du trafic. Cependant, dans notre cadre de VNE multiCloud, l'hébergement intraCloud des applications se fait au niveau de la sélection des DCs. De ce fait, nous avons redéfini les équations incluses dans la fonction multiobjectif et les contraintes du modèle MILP, de manière à rester cohérents avec le niveau de granularité que nous avons pris en compte dans ce travail, qui se limite aux DCs et non aux serveurs physiques logés dans les DCs. L'exécution de la phase d'hébergement intraCloud est nécessaire, notamment pour pouvoir évaluer et valider l'efficacité de la stratégie de partitionnement sur le taux d'acceptation.

3.2.3 Évaluation de performance

À l'aide de simulations numériques, plusieurs expériences ont été conduites dans le but d'évaluer les performances du modèle de partitionnement proposé, notamment sur le taux d'acception, le taux de partitionnement, le délai de communication intraCloud et interCloud, les violations de délai et le temps d'exécution, critères qui sont définis à la Section 3.1.

3.2.3.1 Définition de l'environnement et des outils de simulation

Pour chacun des modèles linéaires ILP et MILP, respectivement pour les phases de partitionnement multiCloud et d'hébergement intraCloud, une méthode exacte de résolution est exécutée. Cette dernière est implémentée en AMPL, en utilisant le solveur (CPLEX, 2018). Toutes les caractéristiques des réseaux sous-jacents et VNRs sont générées aléatoirement, en implémentant un programme MATLAB de génération de jeux de données, avec des valeurs de paramètres uniformément distribuées dans des intervalles définis généralement à partir des références. La topologie des réseaux Cloud est semblable au réseau NSFNet (Amokrane *et al.*, 2015), tandis que celle des VNRs est maillée avec une connectivité aléatoire de 50% entre les VMs. Nous avons également tenu compte du caractère dynamique de l'environnement, en simulant des arrivées de requêtes aléatoires à des intervalles de temps différents et ayant des durées de vie limitées. De ce fait, la méthode de résolution intègre les mises à jour régulièrement faites dans le référentiel d'informations, provenant notamment des ressources allouées après hébergement d'une demande ou libérées après expiration d'une VNR.

3.2.3.2 Évaluation comparative de la stratégie de partitionnement proposée

Pour évaluer notre stratégie de partitionnement, nous avons d'abord défini deux versions de notre approche, à savoir la version complète de notre modèle ILP de partitionnement, et une version 'simplifiée' où aucune contrainte de capacités de ressources VMs et de délai maximal sur les VLs n'est considérée. Nous avons par la suite comparé ces deux versions de notre modèle à trois autres approches citées dans la littérature, en l'occurrence celles proposées par (Dietrich *et al.*, 2015; Leivadeas *et al.*, 2013; Mechtri *et al.*, 2017), en utilisant le même modèle MILP adopté pour la résolution de la phase d'hébergement intraCloud. Dans un premier scénario de simulation, faisant participer 5 CPs dans l'environnement multiCloud et avec des VNRs soumises limitées à 50 VMs, les comparaisons sont faites selon les critères de performance énoncés précédemment. Les résultats montrent d'abord que, pour tous les critères définis, la version complète de notre modèle est plus performante que la version 'simplifiée', ce qui démontre notamment l'importance de considérer les contraintes de capacités des ressources permettant de ne pas surcharger le réseau d'un CP et de dégrader la performance des applications. Ensuite, comparée aux autres approches de référence, notre version complète de stratégie de partitionnement est plus efficace, en résultant en un faible taux de partitionnement, tout en améliorant notamment le taux d'acceptation et le délai de communication interCloud, respectivement de 15.1 %, 18.5 %, et en évitant les violations de la QdS qui sont également réduites de 27.8 %. Dans un second scénario de simulation faisant participer 10 CPs, avec des VNRs soumises limitées à 70 VMs, la complexité du problème

de partitionnement est mise en évidence en évaluant le temps d'exécution. Nous avons arbitrairement fixé un temps limite d'exécution de la méthode exacte de résolution à 21,600 secondes. Les résultats obtenus montrent notamment, qu'au delà de 60 VMs, aucune preuve d'optimalité des solutions générées ne peut être conclue, dès lors que 100% des VNRs ont atteint le temps limite. De ce fait, avec les temps de calcul obtenus croissant de manière exponentielle, une approche approximative de résolution nous est due, afin de pouvoir traiter en particulier de grandes instances du problème en un temps de calcul polynomial.

3.3 Volet 2 : Approche approximative de résolution basée sur la Recherche Taboue ou TS

Les résultats obtenus dans le premier volet du travail ont montré que la méthode exacte de résolution du problème de partitionnement, même si elle a la particularité de produire des solutions optimales, n'est efficace qu'avec des instances de petite taille du problème. En effet, les temps de calcul sont très longs avec l'approche exacte, en raison de la nature NP-Difficile du problème. De ce fait, le but du travail dans ce deuxième volet est de contourner cette complexité, en proposant une approche approximative de résolution basée sur les métaheuristiques. Cette dernière nous permettra de résoudre en un temps de calcul raisonnable le modèle ILP proposé précédemment, en particulier avec des instances de grande taille du problème. Les performances de l'approche heuristique proposée seront évaluées en termes de qualité des solutions obtenues et temps de résolution, où il est question de trouver un excellent compromis entre ces deux critères. Ce but, qui correspond notamment à l'objectif cinq (5) énoncé à la Section 1.3, a été atteint dans l'article intitulé « *A Tabu Search Approach for a Virtual Networks Splitting Strategy Across Multiple Cloud Providers* » (Diallo *et al.*, 2018b), présenté au chapitre 5, dans lequel une approche approximative basée sur TS est proposée pour résoudre le problème de partitionnement des VNRs dans un environnement multiCloud.

3.3.1 Adaptation de la métaheuristique de TS

La métaheuristique de TS est choisie dans notre méthodologie pour son efficacité notoire à résoudre divers problèmes d'optimisation traitant des aspects d'évolutivité, notamment grâce à ses mécanismes de mémoire permettant d'éviter les cycles et les pièges des optima locaux, tel qu'expliqué à Section 2.4.2.2. Notre approche adaptée à l'algorithme de TS, nommée *TS_Split*, combine un processus de recherche local (LS), exécutant entre autres un critère d'aspiration et un mécanisme de mémoire à long terme (Diversification). Afin de déterminer une configuration de solution initiale, plusieurs méthodes ont été testées, à savoir une

méthode gloutonne où chaque VM est assignée au meilleur CP en termes de "quota" d'approvisionnement de ressources et les VLs au plus court chemin entre les CPs hôtes, une autre méthode où toutes les VMs sont assignées à un CP choisi aléatoirement, et une méthode où chaque VM est assignée aléatoirement à un CP répondant à ses attributs fonctionnels, et les VLs assignées à un chemin aléatoire entre les CPs hôtes. Les meilleures solutions ont été obtenues avec la troisième option. De ce fait, *TS_Split* part d'une solution initiale complètement aléatoire et, à chaque itération, le processus de LS parcourt le voisinage de la solution actuelle afin de trouver une meilleure solution. Si aucune amélioration de la meilleure solution trouvée à date n'est observée pendant un certain nombre d'itérations, le mécanisme de Diversification est alors exécuté. Ce dernier consulte une liste statistique ayant mémorisé, depuis le démarrage de l'algorithme, toutes les assignations de VMs aux CPs à partir desquelles l'espace de recherche a été exploré. Par la suite, en se basant sur les zones les moins explorées, une nouvelle solution initiale est construite, à partir de laquelle le processus de LS relance la recherche dans le but de trouver une solution encore meilleure. *TS_Split* s'arrête après un nombre fixe de relances du mécanisme de Diversification. Dans notre démarche méthodologique, deux mouvements nous permettant de passer d'une solution actuelle à une solution voisine ont été examinés, à savoir la permutation entre deux VMs et le déplacement d'une seule VM d'un CP à un autre. Aussi, un mécanisme d'Intensification autour de la meilleure solution trouvée a été également testé. Cependant, les solutions de meilleure qualité ont toujours été obtenues avec le mécanisme de Diversification, et le mouvement simple qui déplace une seule VM à un autre CP était tout aussi efficace que celui de la permutation, et de plus prenait beaucoup moins de temps d'exécution. Par ailleurs, au lieu d'évaluer entièrement chaque configuration de solution obtenue itérativement, ce qui est très couteux en temps de calcul, nous avons défini des fonctions de calcul de gains exprimant la différence entre le coût d'une solution actuelle et celui d'une solution voisine. Ces fonctions de gains incluent le gain associé au coût intrinsèque de la fonction objectif du modèle ILP proposé, ainsi que les gains relatifs aux pénalités liées à la violation de certaines des contraintes associées au problème. Cette procédure de calcul de fonctions de gains est plus compliquée à implémenter, mais elle nous a permis de réduire considérablement l'ordre de complexité du temps de calcul, qui passe de $O(n^2)$ à $O(n)$.

3.3.2 Évaluation de performance

L'approche heuristique proposée est implémentée en C++ et est évaluée par des simulations numériques, en utilisant le même environnement de simulation défini dans le premier volet. Nous avons mené notamment trois phases d'expérimentation, à savoir une première permettant de réaliser le paramétrage de l'algorithme *TS_Split* proposé, une seconde qui a pour

but d'évaluer les performances de *TS_Split* en termes de qualité des solutions obtenues et temps d'exécution, et un dernière permettant d'évaluer les performances de la stratégie de partitionnement des VNRs proposée, selon les mêmes critères que ceux considérés dans le premier volet du travail, mais résolue avec *TS_Split*. Chacune des expériences est réalisée et présentée en fonction de la taille des instances du problème, tel que décrit dans la suite.

3.3.2.1 Scénarios de test

Différents scénarios de test sont examinés, en considérant pour chacun 5 CPs et 10 CPs intervenant dans l'environnement multiCloud. Pour la seconde phase d'expérience, nous avons également étudié les cas simples, où le critère d'aspiration implémenté dans le processus de LS et le mécanisme de Diversification se sont pas exécutés. Pour la troisième phase d'expérience, l'algorithme au complet est exécuté. Pour les instances de petite taille, nous avons considéré les mêmes jeux de données que ceux générés dans le premier volet du travail, avec des VNRs ayant 5 VMs à 70 VMs. Pour les instances de grande taille, nous avons généré des VNRs pouvant contenir 75 VMs à 500 VMs.

3.3.2.2 Tests préliminaires et paramétrage de l'algorithme

Des phases préliminaires de tests sont d'abord effectuées. Ces dernières nous ont permis, en l'occurrence, de pouvoir déterminer la meilleure manière de générer une solution solution initiale et de réaliser des mouvements dans le processus de LS, mais également de définir les bonnes valeurs des paramètres de l'algorithme *TS_Split* en fonction de la taille des instances du problème. Ces paramètres incluent notamment la taille de la liste taboue, le nombre maximal d'itérations sans amélioration de la meilleure solution et le nombre maximal de relances du mécanisme de Diversification. Pour ce qui est du paramétrage de l'algorithme, les expériences ont été réalisées de manière progressive. Pour les instances de petite taille, nous nous sommes basés sur les coûts de solution obtenus avec la méthode exacte exécutée avec AMPL/CPLEX, afin de choisir les valeurs idéales pour les différents paramètres nous permettant d'atteindre ou d'approcher la solution optimale. Pour cela, d'abord la taille idéale de la liste taboue a été déterminée en fixant à des nombre très élevés le nombre d'itérations et le nombre de relances, afin de laisser rouler l'algorithme le plus longtemps possible et d'analyser le comportement et les résultats générés. Par la suite, nous avons fixé le nombre de relances à une valeur très grande afin de déterminer le nombre d'itérations nécessaires à l'algorithme pour s'approcher de la solution exacte, avant de terminer par paramétrer le nombre de relances de la Diversification. Pour les instances de grande taille nous avons procédé avec la même démarche, en s'aidant cette fois-ci de la meilleure solution obtenue

parmi toutes les simulations, et ce pour chaque taille de VNR de 75 à 500 VMs. Des formules mathématiques, en fonction du nombre de VMs dans les VNRs et du nombre de CPs dans le multiCloud, ont pu être dégagées pour chaque paramètre de l'algorithme *TS_Split*, en analysant autant les qualités des solutions obtenues et les temps d'exécution engendrés, dans le but d'arriver à un excellent compromis entre ces deux critères.

3.3.2.3 Comparaison avec la méthode exacte sur des instances de petite taille

Pour les VNRs de petite taille, les tests réalisés montrent une meilleure qualité des solutions générées lorsque le critère d'aspiration est appliqué, et des améliorations considérables avec l'exécution du mécanisme de Diversification. La comparaison avec la méthode exacte montre que l'approche approximative de résolution proposée est capable de générer presque à tous les coups la solution optimale, avec un écart de coût de solution en moyenne à moins de 2.97 % de la solution exacte, et ce en un temps de calcul linéairement réduit. Pour ce qui est d'évaluer les performances de notre stratégie de partitionnement, résolue cette fois-ci avec *TS_Split*, nous avons d'abord implémenté en C++ l'approche de partitionnement basée sur la Recherche Locale Itérée ou ILS proposée par (Leivadeas *et al.*, 2013), nommée *ILS_Split*. Par la suite, nous avons comparé cette dernière à notre approche, en appliquant la même approche multiobjectif précédente de résolution de la phase d'hébergement intraCloud, exécutée de manière optimale avec AMPL/CPLEX. Les résultats prouvent que notre approche heuristique est aussi efficace que la méthode exacte pour prendre les meilleures décisions possibles de partitionnement de VNRs par rapport à *ILS_Split*, en fonction des mêmes critères de performance que ceux considérés dans le premier volet de notre travail.

3.3.2.4 Comparaison avec une méthode de référence sur des instances de grande taille

Pour les VNRs de grande taille, les tests ont montré que sans le mécanisme de Diversification, la qualité des solutions est encore plus dégradée, avec dans certains cas des solutions irréalisables en raison des contraintes de délai maximal violées sur certains VLs. De ce fait, pour de telles instances du problème, la Diversification était incontournable pour l'algorithme proposé. Les résultats obtenus avec les grandes tailles de VNRs confirment les performances de l'algorithme *TS_Split* en termes de qualité de solution, en générant des coûts de solution avec un écart en moyenne de 0.17 % du coût maximal, notamment lorsque le critère d'aspiration est appliqué. Pour ces instances de grande taille, la phase d'hébergement intraCloud est résolue avec un algorithme de descente itérée, et comparé à *ILS_Split*, *TS_Split* donne de meilleurs résultats en un temps de résolution d'exécution également plus réduit.

3.4 Volet 3 : Extension du modèle de partitionnement multiCloud et approche approximative de résolution hybride

Les deux premiers volets du travail s'étaient intéressés à l'optimisation de la performance et de la QdS des applications à héberger dans l'environnement multiCloud. Dans ce troisième volet, il est question d'étendre le modèle ILP proposé pour le partitionnement des VNRs, afin de pouvoir y intégrer les coûts d'approvisionnement à minimiser pour le SP, outre les aspects de performance et de QdS à optimiser simultanément. Par la suite, dans le but de toujours contourner la nature NP-Difficile du problème, une approche approximative de résolution, basée sur une hybridation intégrative des métaheuristiques de colonies de fourmis (ACO) et de TS, est proposée et évaluée. Le nouveau modèle étendu de partitionnement, résolu avec la méthode exacte et la méthode hybride, est ensuite évalué sur des critères de performance et d'économie, en considérant différents scénarios de test. À cet effet, l'approche adoptée pour la résolution de la phase d'hébergement intraCloud des segments de VNRs est également étendue, dans le but de maximiser les profits des CPs sélectionnés, des aspects qui n'étaient pas considérés dans le modèle initial d'hébergement. Ce dernier volet de la thèse nous permet d'atteindre les deux (2) derniers objectifs énoncés à la Section 1.3, à travers l'article intitulé « *An efficient approach based on Ant Colony Optimization and Tabu Search for a resource splitting across multiple cloud providers* », présenté au chapitre 6. Le nouveau modèle mathématique pour la résolution du problème subséquent d'hébergement intraCloud, ainsi que l'algorithme de TS utilisé pour l'approche hybride proposée, sont détaillés dans l'annexe de l'article.

3.4.1 Extension du modèle ILP et minimisation des coûts d'approvisionnement

Le modèle ILP de partitionnement conserve toutes les propriétés du modèle défini dans les deux premiers volets du travail. Cependant, nous avons considéré et modélisé dans le référentiel d'informations les coûts d'approvisionnement de ressources et de services auprès des CPs, que le VNP tentera de minimiser pour le compte du SP. Ces coûts incluent notamment le prix unitaire par type de ressource informatique pour chaque catégorie de VMs,et le prix unitaire de la bande passante pour chaque type de liaison réseau. Chacun de ces prix est fourni par les CPs pour une période de temps donnée. Au niveau du partitionnement des VNRs, pour une communication inter-VMs au sein d'un même CP, le prix unitaire de la bande passante correspond à celui soumis par le CP pour la période de temps. Pour une communication inter-VMs entre deux CPs différents, le prix unitaire de la bande passante publié dans le référentiel accumule celui des deux CPs, ainsi que celui sur chaque liaison interCloud reliant les deux CPs et celui de chaque réseau de transit PoP traversé. Le nouveau

modèle ILP étendu prend en compte la durée de vie des VNRs, et permet au VNP d'optimiser le rapport performance/coût, en sélectionnant les CPs capables d'offrir les meilleures performances et QdS pour les applications à héberger, et ce aux meilleurs prix minimisant le coût total d'approvisionnement de ressources et de services pour le SP.

3.4.2 Hybridation intégrative des métaheuristiques d'ACO et de TS

L'approche de résolution hybride proposée, nommée *ACO_TS_Split*, consiste en une combinaison intégrative des métaheuristiques d'ACO et de TS. Une telle hybridation nous permet de tirer profit des capacités d'adaptation efficaces d'ACO dans le domaine de l'optimisation et de l'efficacité de TS pour éviter les cycles et les pièges d'optima locaux, par les mécanismes de mémoire qu'il inclut. *ACO_TS_Split* utilise une colonie de plusieurs fourmis dont chacune d'elle, à chaque itération, construit une solution initialement vide, en y ajoutant progressivement des composants de solution choisis par l'algorithme d'ACO sur la base d'un modèle probabiliste. Ce dernier est évalué pour chaque composant possible de solution, en utilisant une information heuristique définie à partir du modèle ILP proposé et une valeur de phéromone sur le composant, afin de choisir le meilleur composant à ajouter à la solution à construire. Par la suite, dans la même itération, l'algorithme de TS, qui inclut un processus de LS et un mécanisme de Diversification, est utilisé comme opérateur de LS, afin d'améliorer la meilleure solution construite par les fourmis à l'itération. Le processus de LS est exécuté jusqu'à ce que la solution de la meilleure fourmis locale reste non améliorée pendant un certain nombre d'itérations, avant d'exécuter la Diversification une seule fois selon les même principes que ceux considérés dans le deuxième volet du travail. Par ailleurs, en plus d'utiliser pour la partie TS le même mécanisme d'évaluation accélérée des solutions que celui défini avec *TS_Split* dans le deuxième volet, pour l'algorithme d'ACO également, l'évaluation d'une nouvelle configuration de solution construite par une fourmi est considérablement accélérée. Ceci a été réalisé en utilisant des fonctions de calcul de variations, afin d'uniquement évaluer la différence de coût générée par l'ajout d'un nouveau composant à la configuration partielle d'une solution actuelle, ce qui réduit nettement le temps d'exécution de l'algorithme hybride

3.4.3 Évaluation de performance

L'approche hybride proposée est implémentée en C++ et est évaluée par des simulations numériques, en utilisant le même environnement de simulation défini dans le premier volet. Dans cette troisième partie du travail, nous avons également mené trois phases d'expérimentation, consistant en une première permettant de réaliser notamment le paramétrage de l'algorithme *ACO_TS_Split* proposé, aussi bien pour la partie d'ACO que celle de TS. Par la suite, une

seconde phase ayant pour but d'évaluer les performances de *ACO_TS_Split* en termes de qualité des solutions obtenues et temps d'exécution selon la taille des instances du problème est élaborée, avant de réaliser une dernière expérimentation permettant d'évaluer la nouvelle stratégie de partitionnement des VNRs proposée, selon certains critères de performance et d'économie.

3.4.3.1 Scénarios de test

Les différents scénarios de test sont définis en considérant 10 CPs intervenant dans l'environnement multiCloud. Les deux premières phases d'expérimentation ont été réalisées en fonction de la taille des instances des VNRs, qui sont composées de 5 à 70 VMs pour les petites tailles, et de 75 à 500 VMs pour les grandes tailles. La troisième phase d'expérimentation n'est réalisée que sur les instances de petite taille. Dans la deuxième phase d'expérience, nous avons comparé l'approche hybride proposée, avec Diversification et sans Diversification, à l'approche exacte exécutée avec AMPL/CPLEX, à l'algorithme d'ACO simple sans TS comme LS, et à l'algorithme *TS_Split* correspondant à celui développé dans le deuxième volet du travail. Dans la troisième phase d'expérience, les performances de la nouvelle stratégie de partitionnement sont comparées à celles du modèle stratégique où seules la performance et la QdS des applications sont optimisées, comme c'est le cas avec les deux premiers volets du travail, et à celles du modèle stratégique où seuls les coûts d'approvisionnement pour le SP sont minimisés, similairement au modèle de partitionnement proposé par (Dietrich *et al.*, 2015). Toutes ces différentes stratégies de partitionnement sont résolues dans cette troisième étape d'expérimentation avec l'approche hybride *ACO_TS_Split* complète, et la phase d'hébergement intraCloud maximisant les profits du CP est résolue de manière optimale avec la méthode exacte, exécutée avec AMPL/CPLEX.

3.4.3.2 Tests préliminaires et paramétrage de l'algorithme

Des phases préliminaires de tests sont également effectuées afin de définir les bonnes valeurs des paramètres de l'algorithme hybride *ACO_TS_Split*, en fonction de la taille des instances du problème. Le paramétrage est réalisé en définissant progressivement les valeurs idéales pour le taux d'évaporation de phéromone, l'importance relative du taux de phéromone et de l'information heuristique dans le modèle probabiliste, la taille de la colonie de fourmis et le nombre maximal d'itérations de *ACO_TS_Split*. Le paramétrage de l'algorithme de TS est ensuite réalisé dans une même démarche que celle dans le deuxième volet du travail. Pour les instances de petite taille, nous avons utilisé les solutions obtenues avec la méthode exacte pour réaliser le paramétrage. Pour celles de grande taille, les meilleures solutions

générées parmi toutes les simulations pour les différentes instances sont utilisées. Des formules mathématiques en fonction du nombre de VMs dans les VNRs et du nombre de CPs dans le multiCloud sont alors définies pour les différents paramètres de l'algorithme *ACO_TS_Split*, dans le but de toujours arriver à un excellent compromis entre la qualité des solutions générées et le temps de résolution.

3.4.3.3 Évaluation des performances de l'approche de résolution hybride

Pour les VNRs de petite taille, la comparaison avec les méthodes exacte, d'ACO simple et *TS_Split*, montre que l'approche hybride *ACO_TS_Split*, incluant le mécanisme de Diversification, améliore considérablement la qualité des solutions obtenues par rapport à l'algorithme d'ACO simple ou à l'algorithme *ACO_TS_Split* sans Diversification, mais au prix d'un temps de résolution relativement plus élevé. L'algorithme *TS_Split* donne des solutions satisfaisantes mais en moyenne légèrement de moins bonne qualité que *ACO_TS_Split* avec Diversification et ce en un temps de résolution un peu plus élevé. En particulier avec des VNRs de petite taille, les deux algorithmes *ACO_TS_Split* avec Diversification et *TS_Split*, génèrent des solutions très proches de la solution exacte, avec un écart de coût moyen à moins de 3.42 % et 4.18 % respectivement. Avec des VNRs de grande taille, les comparaisons ne sont faites qu'avec *TS_Split*, car l'algorithme d'ACO simple donne dans certains cas des solutions irréalisables. Des résultats similaires sont observés entre *ACO_TS_Split* et *TS_Split*, les solutions étant en moyenne meilleures avec l'approche hybride complète qu'avec l'algorithme de TS.

3.4.3.4 Évaluation du modèle étendu de partitionnement sur les critères de performance et d'économie

La nouvelle stratégie étendant le modèle ILP de partitionnement est évaluée sur le taux d'acceptation des VNRs, le coût d'approvisionnement pour le SP et le profit du CP. Les résultats montrent des taux d'acceptation plus bas dans les cas où uniquement la réduction des coûts est l'objectif principal du SP (ou alors qu'aucune information sur les ressources n'est divulguée au VNP, à l'exception du prix unitaire). Cependant le taux d'acceptation des VNRs est en moyenne meilleur dans les cas où seule l'optimisation des performances des VNRs est prise en compte dans la stratégie de partitionnement. Par ailleurs, le nouveau cadre de VNE multiCloud étendu améliore le rapport performance/coût d'environ 8 % pour le SP, et le profit d'environ 67.2 % pour les CPs.

3.5 Synthèse sur les travaux

En synthèse, l'ensemble des travaux réalisés dans cette thèse et présentés dans les trois chapitres subséquents, nous a permis d'atteindre les différents objectifs énoncés à la Section 1.3. Le cadre de VNE multiCloud proposé permettra d'offrir au SP un bon compromis entre la performance de ses applications à héberger dans les différentes infrastructures Cloud et le coût d'approvisionnement induit, et de faciliter ses choix de services Cloud selon ses besoins et ses intérêts. Les approches de résolution approximatives proposées permettent largement de faire face à la complexité du problème, en offrant un excellent compromis entre la qualité des solutions générées et le temps d'exécution. Les CPs intervenant dans l'environnement pourraient également y voir un moyen de publier davantage d'informations sur les ressources et les services offerts, autres que leurs coûts monétaires uniquement, afin d'augmenter le taux d'acceptation, et par conséquent les profits.

CHAPITRE 4 ARTICLE 1 : A QOS-BASED SPLITTING STRATEGY FOR A RESOURCE EMBEDDING ACROSS MULTIPLE CLOUD PROVIDERS

Auteurs : Marieme Diallo, Alejandro Quintero et Samuel Pierre.
Revue : Accepté pour publication dans le journal *IEEE Transactions on Services Computing*, en Octobre 2018.

Abstract

In cloud computing, a fundamental management problem with the Infrastructure as a Service model lies in the efficient embedding of computing and networking resources onto distributed virtualized infrastructures. This issue, usually referred to as the Virtual Network Embedding (VNE) problem, has been well studied for a single Cloud Provider (CP). However, wide-area services delivery may require to embed heterogeneous resources over multiple CPs. This adds more complexity and scalability issues, since the Virtual Network Requests (VNRs) embedding process requires two phases of operation: the multicloud VNRs splitting, followed by the intracloud VNR segments mapping. This paper addresses the problem of VNE across multiple CPs by proposing a VNRs splitting strategy which aims at improving the performance and QoS of resulting VNR segments. An Integer Linear Program (ILP) is used to formalize the splitting phase as a maximization problem with constraints. Subsequently, in order to minimize the overall delay, a multi-objective intracloud resource mapping approach formalized as a Mixed-Integer Linear Program (MILP) is adopted. Simulations with the exact method show the efficiency of the proposed strategy based on several performance criteria. In particular, the acceptance rate and the delay are respectively improved by 15.1% and 18.5%, while preventing QoS violations.

4.1 Introduction

Over the last few years, with the success of the Internet and the rapid development of data processing and storage technologies, cloud computing has emerged as an innovative utility computing model (Armbrust *et al.*, 2010; Zhang *et al.*, 2016a). In this computing paradigm, customers can access a large pool of distributed configurable resources as an on-demand service model via the Internet. From small businesses perspective, it is widely approved as a large-scale economic model, avoiding huge investments in operation and maintenance of hardware and software. Among all the cloud service models defined by the National

Institute of Standards and Technology (Mell et Grance, 2009), the Infrastructure as a Service (IaaS) has nowadays become the most widely adopted in cloud computing (Manvi et Shyam, 2014). In this new business model, the traditional role of the Internet Service Provider (ISP) is decoupled into two main new roles (Rabah *et al.*, 2015; Fischer *et al.*, 2013): the infrastructure provider or Cloud Provider (CP) and the Service Provider (SP). The CP uses virtualization technologies to provide a pool of infrastructure layer resources to the SP, including processing, storage, network access and routing services. The CP is then responsible for managing and maintaining the virtualized substrate infrastructure network. The SP is in charge of deploying various protocols, operating systems and applications, packaged into connected Virtual Machines (VMs) (Chaisiri *et al.*, 2012). Thus, the SP can lease virtual resources (computing and networking) from one or more CPs, in order to build heterogeneous virtual networks that will offer end-to-end customized services to end users. Virtual Network Requests (VNRs) are submitted in a high level of abstraction by SPs, as a set of virtual nodes (e.g. VMs) interconnected by virtual links (VLs) representing the exchanged traffic between the virtual nodes.

In the IaaS-based cloud model, the fundamental challenge is how to efficiently embed co-existing VNRs onto distributed substrate infrastructures, including constraints on resource allocation and QoS (Zhang *et al.*, 2016a; Manvi et Shyam, 2014). In the concept of network virtualization, this issue is usually referred to as the NP-hard Virtual Network Embedding (VNE) problem (Zhang *et al.*, 2016a; Fischer *et al.*, 2013), (Chowdhury *et al.*, 2012). The latter deals with the embedding of VNRs, where heterogenous resources are allocated to host virtual nodes (e.g. VMs) in specific substrate nodes (e.g. data centers), and to route VLs onto substrate paths. VNE over a single-cloud network has been well addressed in past research works (Chowdhury *et al.*, 2012; Melo *et al.*, 2013, 2015; Larumbe et Sanso, 2013; Houidi *et al.*, 2015; Zhang *et al.*, 2014; Amokrane *et al.*, 2015; Hesselbach *et al.*, 2016; Ayoubi *et al.*, 2016; Shahriar *et al.*, 2017; Khan *et al.*, 2016; Cao *et al.*, 2017; Pyoung et Baek, 2018). However, wide-area services deployment from one geographical location to another may require to embed VNRs across heterogeneous substrate infrastructures owned by multiple CPs (Grozev et Buyya, 2012; Rafael *et al.*, 2012).

VNE over a multicloud network has been only recently addressed in the literature (Houidi *et al.*, 2011; Samuel *et al.*, 2013; Leivadeas *et al.*, 2013; Gong *et al.*, 2016; Li *et al.*, 2016; Mechtri *et al.*, 2017; Dietrich *et al.*, 2015; Mano *et al.*, 2016; Ran *et al.*, 2016; Aral et Ovatman, 2016). In this context, a third actor, called the Virtual Network Provider (VNP), intervenes between the SP and the CPs, acting as a virtual brokerage service on behalf of the SP (Rabah *et al.*, 2015; Fischer *et al.*, 2013). The entire embedding process then requires two major phases of operation with different purposes and technical approaches: the splitting

phase, followed by the mapping phase. In the first phase, the VNP uses a strategy to select eligible CPs based on SP's requirements, and partitions the VNRs into segments. In the second phase, each selected CP uses a single-cloud VNE approach to map the assigned VNR segments into its intracloud network. The problem is then more challenging than the one commonly known with a single CP, particularly with the splitting phase. The latter deals with the NP-complete multiway separator problem (Sanchis, 1989; Tao *et al.*, 1992). On the other hand, solutions with the single-cloud VNE problem can be optimal since they are based on a complete knowledge of the substrate network topologies and how the available resources are distributed. In the context of multicloud VNE, such information is concealed by the CPs during the splitting phase (Dietrich *et al.*, 2015; Mano *et al.*, 2016). This leads the VNP to proceed with the splitting phase based on a very restricted knowledge of the multicloud environment. Moreover, the intra-domain mapping situation of a VNR segment assigned to a particular CP is not shared with the other CPs. Therefore, solutions with the multicloud VNE problem may not be optimal from the global view of the embedding process. An efficient splitting strategy should take into account CP policies that limit information that can be accessed by the VNP, while generating embedding solutions which best satisfy SP's requirements. However, the few studies on the multicloud VNE problem have generally assumed that CPs would disclose some private information (Leivadeas *et al.*, 2013; Samuel *et al.*, 2013). Some others aim only at minimizing the resource provisioning price for the SP during the splitting phase (Dietrich *et al.*, 2015; Mano *et al.*, 2016; Gong *et al.*, 2016; Ran *et al.*, 2016; Samuel *et al.*, 2013; Houidi *et al.*, 2011), regardless of the performance and QoS of the resulting embedded VNR segments.

In order to address the challenges above-mentioned, we propose in this paper a QoS-based splitting strategy for embedding VNRs across multiple IaaS providers. Our approach considers resource and QoS constraints based on SP's requirements, with the purpose of efficiently splitting the VNRs and improving the performance of resulting VNR segments that are mapped onto the selected intracloud infrastructures. The key contributions of this work are as follows:

— Our optimization strategy considers jointly VMs and inter-VMs traffic in the embedding process, by taking into account the required computational and networking resources as well as the QoS requirements specified by the SP;

— We have investigated the restricted access level of the VNP on the substrate network and designed a multicloud VNRs splitting approach based on the limited information disclosed by the CPs. In addition, for a better efficiency of our splitting strategy, we have enriched the visibility of the VNP on the multicloud environment. This is achieved by taking advantage of certain information related to network topologies

that is not treated as confidential, such as Points-of-Presence (PoPs) of CPs (Dietrich *et al.*, 2015; Li *et al.*, 2016; Gong *et al.*, 2016), including some performance guarantees for a link (e.g. delay bounds) (Gong *et al.*, 2016; Larumbe et Sanso, 2013);

— We use an Integer Linear Programming (ILP) model to formalize the VNRs splitting phase as a maximization problem with constraints. Resource provisioning costs are defined and weighted based on the availability of supplied resources and the performance guarantees advertised to the VNP;

The rest of the paper is organized as follows: Section 4.2 discusses related work. Section 4.3 describes the multicloud VNE problem and presents the proposed embedding framework. Section 4.4 presents the system modeling and problem formulation related to the proposed VNRs splitting strategy. Performance evaluation and numerical results are presented in Section 4.5. Finally, conclusion and future works are highlighted in Section 4.6.

4.2 Related work

VNE represents the central resource allocation challenge in virtualized cloud infrastructures (Zhang *et al.*, 2016a; Manvi et Shyam, 2014). It is known to be an NP-hard problem (Fischer *et al.*, 2013), (Chowdhury *et al.*, 2012). If we need to only embed non-communicating virtual nodes, the problem is generally reduced to the well-known NP-hard Bin packing problem (Zhang *et al.*, 2016a). In the case of interconnected virtual nodes, the problem leads to a simultaneous mapping with constraints of virtual nodes and VLs onto the substrate networks. The problem is then reduced to the NP-hard multiway separator problem (Fischer *et al.*, 2013; Sanchis, 1989; Tao *et al.*, 1992). Even when all virtual nodes are already mapped, the VLs embedding problem remains NP-hard (Chowdhury *et al.*, 2012). As a result, various studies have proposed optimized exact or heuristic VNE algorithms, in order to satisfy economic benefits, resource utilization-efficiency, energy-efficiency, survivability and QoS aspects.

Here we discuss related work on VNE over a single-cloud network and VNE over a multicloud network.

4.2.1 VNE over a single-cloud network

Many algorithms, mostly based on (meta) heuristic approaches, have been proposed for single-domain VNE optimization problems (Pyoung et Baek, 2018; Cao *et al.*, 2017; Hesselbach *et al.*, 2016; Khan *et al.*, 2016; Ayoubi *et al.*, 2016; Amokrane *et al.*, 2015; Chowdhury *et al.*, 2012; Shahriar *et al.*, 2017; Zhang *et al.*, 2014; Larumbe et Sanso, 2013; Houidi *et al.*, 2015; Melo *et al.*, 2013, 2015; Ghaleb *et al.*, 2016). Some works tend to solve the VNE problem

by providing a certain coordination between the node mapping stage and the link mapping stage (Chowdhury *et al.*, 2012; Hesselbach *et al.*, 2016). (Chowdhury *et al.*, 2012) proposed a mixed-integer programming (MIP) in a two-step mapping approach with better coordination between node and link mappings. In the same way, (Hesselbach *et al.*, 2016) recently proposed a new path algebra-based embedding strategy coordinating in a single step the mapping of nodes and links. Their optimization strategy has significantly improved the performance of the proposed algorithms.

Dynamic methods, which support remapping of resources during the lifetime of requests, have also been proposed (e.g.(Zhang *et al.*, 2014; Ayoubi *et al.*, 2016; Khan *et al.*, 2016; Shahriar *et al.*, 2017)). Zhang *et al.* (Zhang *et al.*, 2014) introduced a novel approach to support opportunistic resource sharing among virtual nodes and VLs. The authors propose an ILP-based algorithm which perform resource re-optimization and re-embedding, by considering the dynamic time-varying of both resource requirements on the virtual network and the resource availability on the substrate network. (Ayoubi *et al.*, 2016) took a step further by proposing a Tabu-based framework for reliable VNE with migration, which consists in availability-aware resource allocation and reconfiguration. Shahriar *et al.* (Shahriar *et al.*, 2017) aimed at network survivability, by performing re-embeddings in case of failures in the substrate network. Ayoubi *et al.* Because most of these approaches may suffer from improper load balancing and link underutilization, Khan *et al.* (Khan *et al.*, 2016) proposed a multi-path link embedding strategy by exploiting the path splitting capability to achieve VNE survivability, while minimizing both resource redundancy and path splitting overhead.

Other works introduce multi-objective approaches to resolve the problem (Pyoung et Baek, 2018; Cao *et al.*, 2017; Larumbe et Sanso, 2013; Melo *et al.*, 2013, 2015; Houidi *et al.*, 2015). (Larumbe et Sanso, 2013) proposed an efficient multi-objective MILP model for mapping virtual resources into various locations of cloud data centers. Their solution allows CPs to jointly minimize delay, environmental and resource operating costs. In the same way, the authors (Melo *et al.*, 2013) proposed an ILP model to solve the online VNE problem in order to minimize the resource consumption, while performing load balancing. Their work was extended in (Melo *et al.*, 2015) by considering the minimization of energy consumption. (Houidi *et al.*, 2015) also proposed a multi-objective fault-tolerant VNE algorithm formalized as a MILP. Their approach takes into account constraints related to power consumption, resource availability and load balancing. (Pyoung et Baek, 2018) recently propose an embedding framework considering jointly load balancing and energy saving so as the CP's profit is maximized.

These intra-domain VNE solutions, although they may be optimal, are based on a complete

knowledge of substrate network topologies and available resource distributions. As a result, they can not be properly applicable to a multi-domain VNE problem. On the other hand, upon splitting the VNRs, they can be used by each CP for the VNR segments intracloud mapping phase.

4.2.2 VNE over a multicloud network

VNE over a multi-domain has not been intensively studied. It has been mainly addressed by authors (Mechtri *et al.*, 2017; Dietrich *et al.*, 2015; Leivadeas *et al.*, 2013; Mano *et al.*, 2016; Gong *et al.*, 2016; Li *et al.*, 2016; Ran *et al.*, 2016; Aral et Ovatman, 2016; Samuel *et al.*, 2013; Houidi *et al.*, 2011). Research works essentially focused on the VNRs splitting phase, since existing solutions related to the single-domain VNE problem are usually adopted for the mapping phase.

(Houidi *et al.*, 2011) were one of the first authors working in this research field. They proposed an approach that compares exact and heuristic methods in terms of VNE efficiency, by using both ILP and a max-flow min-cut algorithm for the VNRs splitting. (Samuel *et al.*, 2013) introduced a distributed protocol for VNE problem in multiple substrate networks. Their approach aims at coordinating the participating CPs through competitive pricing mechanisms, in order to maximize their revenue.

The proposals in (Mechtri *et al.*, 2017; Leivadeas *et al.*, 2013; Li *et al.*, 2016; Aral et Ovatman, 2016) are among the few performance-based approaches for VNRs splitting. (Leivadeas *et al.*, 2013) proposed a hierarchical framework that handles the VNRs splitting through an ILP model where resource provisioning costs are defined based on the scarcity and the average utilization. However, their approach does not respect certain information concealed by the CPs (Dietrich *et al.*, 2015; Mano *et al.*, 2016). Moreover, they defined the scarcity as the number of matching substrate nodes and substrate links a CP can offer, which may not necessarily reflect the performance of the CP's substrate network. (Li *et al.*, 2016) propose a multi-domain link mapping framework which jointly considers the mapping of intra and peering links so as the overall resource utilization is optimized. (Mechtri *et al.*, 2017) recently propose a generic ILP model based on a binary distance metric, which eliminates unfeasible solutions in terms of resource and delay violations. This distance measures the closeness between the requested and the selected resources (CPU, memory, storage), as well as the closeness between the delay on a substrate path and the latency requested on a VL that the path has not to exceed.

On the other hand, some works investigated the level of access to information of the VNP before proceeding to the splitting phase (Dietrich *et al.*, 2015; Mano *et al.*, 2016; Gong

et al., 2016). (Gong *et al.*, 2016) and (Dietrich *et al.*, 2015) investigated the feasibility of multi-domain VNE with limited information disclosure (LID) and use the location of peering nodes and peering links of CPs to propose an ILP splitting model which minimizes the total resource provisioning expenditure for the SP. The authors (Mano *et al.*, 2016) proposed also a novel optimization method restricted to CPs' private information, by using minimal MPC operations that minimize inter-domain VNE prices. However, these proposals are only focus on VNRs splitting at the lowest price for the SP, without considering performance and QoS of embedded VNR segments.

Based on the literature, most strategies related to the VNRs splitting across multiple CPs aim only at minimizing the splitting expenditure (Dietrich *et al.*, 2015; Gong *et al.*, 2016; Ran *et al.*, 2016; Mano *et al.*, 2016; Houidi *et al.*, 2011). Not to mention that some proposals that focus on performance-based approaches take into account only limited types of resources or QoS specifications in the scenarios (Mechtri *et al.*, 2017; Leivadeas *et al.*, 2013; Li *et al.*, 2016; Aral et Ovatman, 2016). Some others assume that CPs would share some private information with the VNP (Leivadeas *et al.*, 2013; Samuel *et al.*, 2013), or those investigating the domain-privacy of providers (Dietrich *et al.*, 2015; Mano *et al.*, 2016; Gong *et al.*, 2016) tend only to minimize the resource provisioning price for the SP, which may not give opportunities to select CPs based on desired performance and QoS.

4.3 Multicloud VNE problem

In this section, we first describe challenging aspects related to the VNE across multiple CPs. Then, we present in details the proposed multicloud VNE framework, where a hierarchical approach is generally adopted to decompose the global problem into the VNRs splitting phase and the VNR segments mapping phase.

4.3.1 Problem description

The global multicloud VNE process is described in Figure 4.1, which consists in mapping VNRs onto one or multiple cloud networks. VNRs are formulated by the SP as a set of virtual nodes (e.g. VMs), interconnected by VLs representing the exchanged traffic between virtual nodes (Fischer *et al.*, 2013). Virtual nodes and VLs are specified with criteria and constraints related to the required resources (e.g., processing, memory, storage, bandwidth, QoS parameters, geographic footprint, etc.), that the embedding process has to satisfy. In the case of VNE over a single CP, the SP usually communicates directly with the CP, which can proceed to an optimal VNRs intracloud mapping by exploiting the complete knowledge of

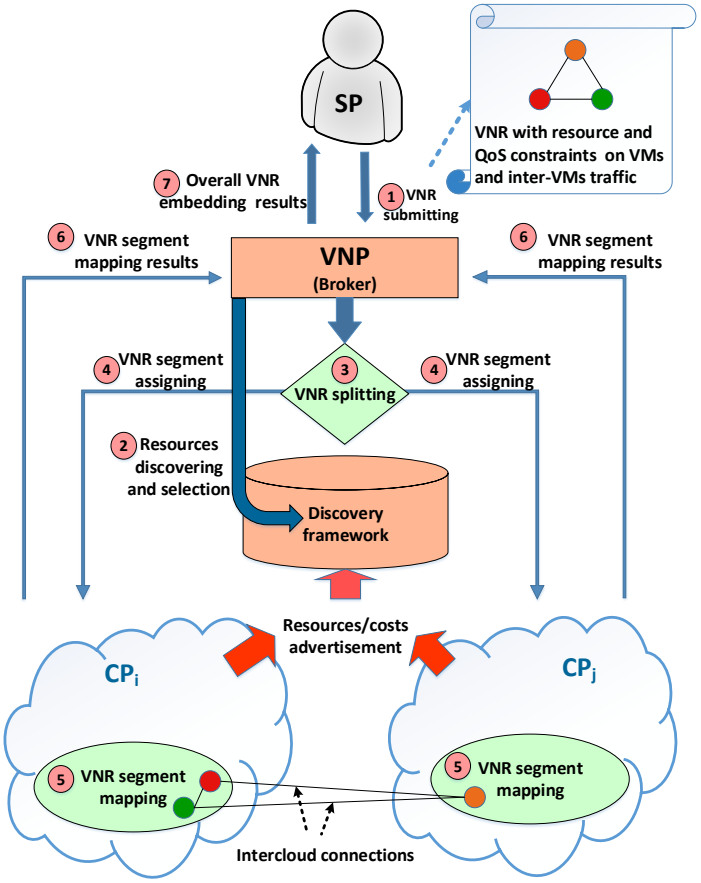

Figure 4.1 Multicloud VNE process.

its substrate network. But, in the context of VNE across multiple CPs, the SP will generally rely on the VNP. The latter acts as a broker for discovering and selecting a set of advertised resources, collected from multiple CPs and stored in a resource discovery framework (Rabah *et al.*, 2015). If there is no single CP able to satisfy all the SP's requirements, the VNP will need to efficiently split the VNRs among eligible CPs. The resulting VNR segments are then mapped by each CP to which they are assigned, and are subsequently interconnected via appropriate intercloud links (Leivadeas *et al.*, 2013).

The entire multicloud VNE process then deals with an NP-hard problem for each of the VNRs splitting and VNR segments mapping phases (Zhang *et al.*, 2016a; Fischer *et al.*, 2013). This adds more challenges since the global problem requires two different technical approaches with different purposes to resolve it. In addition, the VNP visibility on the multicloud substrate network is essentially decisive for the efficiency of any VNRs splitting strategy. However, the VNP proceeds to the splitting phase with a very limited knowledge on the multicloud environment (Dietrich *et al.*, 2015). Indeed, as illustrated in Figure 4.2 (a),

detailed information such as substrate network topologies, router-level connectivity, transit network topologies, number of data centers (DCs) located in a cloud and their interconnectivity, number of instances of available resources and their utilization, are not disclosed to the VNP.

Moreover, CP policies restrict the interoperability with other parties, leading each CP to map a particular segment of VNR without any information of the mapping situation of the rest of the VNR. As a result, solutions with the multicloud VNE problem may not be optimal from the global view of the entire embedding process. An effective splitting strategy should conduct a thorough investigation of the resource information discovery, and so, can avoid inefficient embedding solutions while complying with CP policies.

4.3.2 Multicloud provider VNE framework

Here, an analysis of the resource discovery framework is presented, followed by the description of each of the splitting and mapping phases.

4.3.2.1 Resource discovery framework and assumptions

A CP typically classifies its substrate nodes into different categories (e.g. (Amazon, 2017)), each of them having common functional attributes and associated with a set of computational resource types. Functional attributes define characteristics and properties related to the node type, operating system, virtualization environment, geographic footprint, QoS parameters, etc. Computational resources can be the CPU, disk space, memory, etc., each of them associated with a capacity that the CP keeps dynamically updated (Houidi *et al.*, 2011). Similarly, substrate links are also classified into different types (e.g. VLAN, L3/L2 VPN, etc.) and are associated with a bandwidth capacity dynamically updated too.

However, as previously mentioned, the VNP has no information on intra-domain topologies of CPs, neither on residual capacities of each instance of resource. Nevertheless, some information about transit networks (PoPs) of CPs can be accessible in a high level of abstraction (Doverspike *et al.*, 2010), allowing the VNP to expand its limited view on the substrate multicloud network (Dietrich *et al.*, 2015; Li *et al.*, 2016). Those information can include an oversimplified view of PoP networks, traffic statistics and some performance guarantees for a link such as delay bounds (Gong *et al.*, 2016; Larumbe et Sanso, 2013). A CP can also supply the maximum capacity of computational resources it can offer, or the minimum bandwidth capacity it can guarantee (Chaisiri *et al.*, 2012), without revealing details about the distribution and utilization of these resources within its intracloud network.

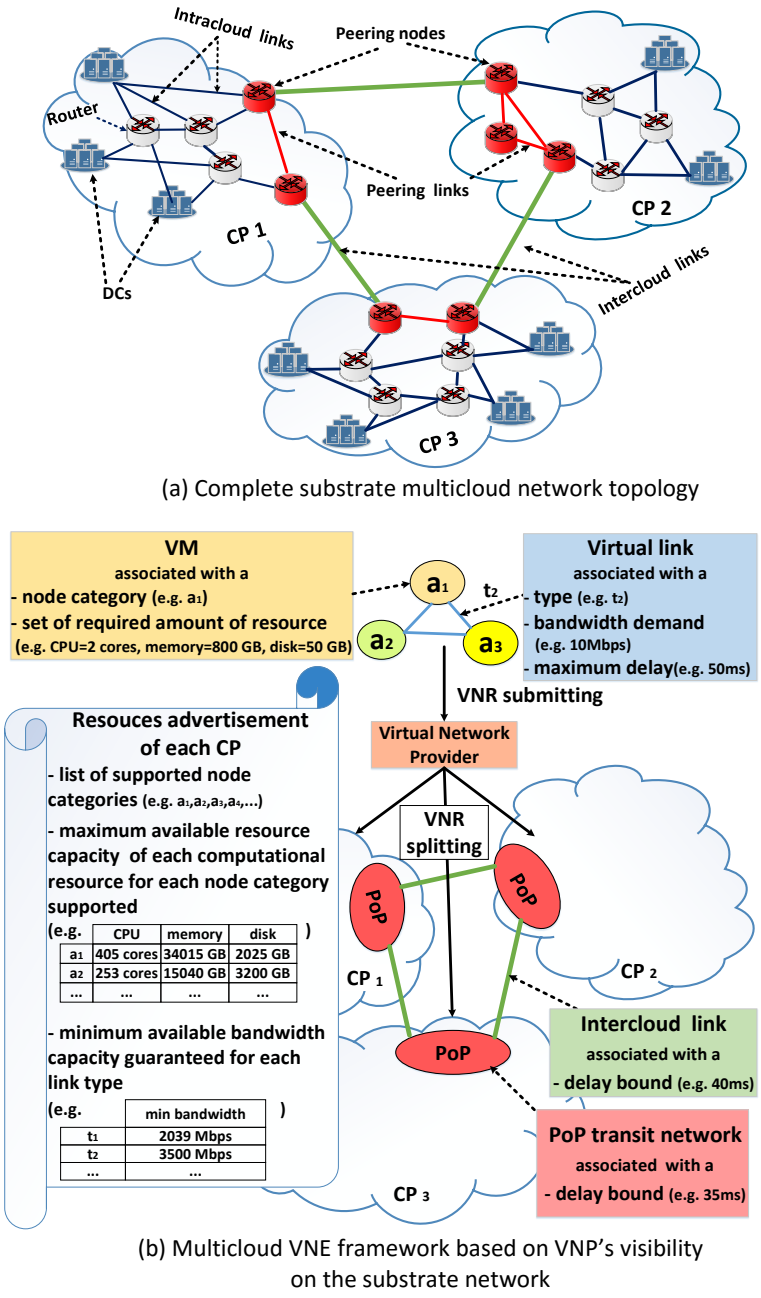

Figure 4.2 Substrate multicloud network

Based on these observations, we consider in our framework the following information accessible by the VNP to process to the VNRs splitting: (i) the maximum available capacity of each computational resource type a CP can offer for each category of node, (ii) the minimum available bandwidth capacity a CP can guarantee in its intra-domain for each type of link, (iii) the estimated delay bound on an intercloud link and for transiting by a PoP network. Figure 4.2.b describes our multicloud provider VNE framework for the splitting phase.

Since the internal topology of PoP networks is not disclosed to the VNP, a PoP network is represented in a high level of abstraction. Peering nodes and peering links are aggregated into a single macro node, which is associated with a delay bound for transiting through it. For sake of simplicity, we only consider one PoP network per CP. We did not consider in this work the path splitting scenario, as the VNP does not know the residual bandwidth capacity of intercloud paths. Without loss of generality, we consider that virtual nodes are virtual machines (VMs) packaged by the SP. Note that CPs may also advertise the unit monetary cost of resources they offer in the discovery framework. But, in this present work, the expenditure minimization for the SP is not our interest. We only focus on an efficient VNE that targets the performance and the QoS of split VNRs. To this end, our strategy handles the VNRs splitting as a maximization problem, where resource provisioning costs are defined based on the information listed above, with some QoS constraints that have been taken into account.

4.3.2.2 Multicloud VNRs splitting phase

Upon receiving an incoming VNR, the VNP relies on the discovery framework to identify eligible substrate resources able to fulfill the SP's requirements. To simplify this matching step, we assume that VMs and VLs are specified with characteristics at the same level as substrate nodes and substrate links respectively. As shown in Figure 4.2 (b), each VM is classified into a specific category of node (e.g. a_1) which includes the desired location, and is defined with a set of computational resource types (i.e. CPU, disk space, memory), each of them associated with the corresponding amount of resource required. Similarly, each VL is associated with a type (e.g. t_1), as well as an amount of bandwidth demand and a maximum delay allowed on the link.

Resources required by a VNR may be supplied by more than one CP, at different provisioning costs, depending on their location and their availability. In our framework, the splitting phase is performed by using an ILP model (detailed in Section 4.4.3), which evaluates the most cost-effective resource provisioning among several candidate CPs, while considering QoS restrictions specified by the SP.

4.3.2.3 Intracloud VNR segments mapping phase

The intracloud VNR segments mapping phase corresponds to the well-known VNE problem with a single CP. Each CP that receives a particular VNR segment maps it onto its infrastructure by using an appropriate intra-domain mapping method. In our framework, we resolve the mapping phase problem by adopting an efficient multi-criteria solution based on

the principles stated by F. Larumbe and B. Sanso (Larumbe et Sanso, 2013). In their approach, a selected CP must map in the best case all the assigned request segments, by using a multi-objective MILP model that aims to minimize the total intracloud embedding cost (including resource, traffic and environmental costs), while providing the best possible QoS by reducing mainly the overall traffic routing delay. However, we have re-stated the adopted approach in order to remain consistent with the level of granularity we have considered in this work, which is limited to DCs and not to physical servers hosted into the DCs.

4.4 System model and formulation

In this section, the respective modelings of the multicloud substrate network and the VNR are presented, followed by the mathematical formulation related to the VNRs splitting problem. An ILP model is then proposed to describe our QoS-based VNRs splitting strategy. Table 4.1 summarizes all the notation we use in our model. The modeling, notation and formulation related to the intra-domain VNR segments mapping problem are available in Appendix A.

4.4.1 Multicloud substrate network modeling

The multicloud substrate network modeling is based on the VNP's network topology visibility, as illustrated in Figure 4.2 (b). The substrate network is modeled as a weighted undirected graph $G^S = (N^S, L^S)$, where N^S is the set of substrate nodes and L^S the set of substrate links (intercloud links). Let I designate the set of CPs and Θ the set of all PoP nodes (transit networks). Note that N^S includes I and Θ, and PoP nodes are only used to access or transit the network of a CP. Let A designate the set of node categories and R the set of computational resource types. The maximum available capacity of resource $r \in R$ that CP $i \in I$ can offer for nodes of category $a \in A$ is denoted by Q_{rai}. Similarly, let T designate the set of link types. The minimum available bandwidth capacity that CP $i \in I$ can guarantee in its intracloud network for links of type $t \in T$ is denoted by B_{ti}. The average delay bound on intercloud link $e \in L^S$ and for transiting through PoP network $p \in \Theta$ are respectively denoted by d_e and d_p. Let $\mathcal{P}_{ij}$ designate the set of all paths between CP i and CP j. The average delay bound on path $\varphi \in \mathcal{P}_{ij}$ is denoted by d_φ. We denote by $\mathcal{O}_\varphi$ the set of all PoP transit networks in Θ spanned by path $\varphi \in \mathcal{P}_{ij}$.

Table 4.1 Notation for the VNRs splitting problem

Symbols	Description
	Global sets
A	Set of node categories
T	Set of link types
R	Set of computational resource types
	Multicloud substrate network
G^S	Graph representing the multicloud substrate network
I	Set of CPs
Θ	Set of PoP transit networks
N^S	Set of substrate nodes in G^S $(N^S = I \bigcup \Theta)$
L^S	Set of substrate links in G^S
Q_{rai}	Maximum available capacity of resource $r \in R$ that CP $i \in I$ can offer for nodes of category $a \in A, (Q_{rai} \in \mathbb{N})$
Q^M_{ra}	Highest available capacity of resource $r \in R$ supplied among all CPs for nodes of category $a \in A$
B_{ti}	Minimum available bandwidth capacity that CP $i \in I$ can guarantee for links of type $t \in T,\ (B_{ti} \in \mathbb{N})$
B^M_t	Highest available bandwidth capacity guaranteed among all CPs for links of type $t \in T$
d_e	Average delay bound on link $e \in L^S$
d_p	Average delay bound for transiting through PoP $p \in \Theta$
d_φ	Average delay bound on path $\varphi \in \mathcal{P}_{ij},\ i, j \in I$
$\mathcal{P}_{ij}$	Set of all paths between CP i and CP j
$\mathcal{O}_\varphi$	Set of PoP transit networks in Θ spanned by path $\varphi \in \mathcal{P}_{ij},\ i, j \in I$
	Virtual network request
G^V	Graph representing the VNR
N^V	Set of VMs in G^V
L^V	Set of VLs in G^V
N^V_a	Set of VMs of category of node $a \in A$
L^V_t	Set of VLs of type $t \in T$
q_{rv}	Amount of resource $r \in R$ required by VM $v \in N^V$ $(q_{rv} \in \mathbb{N}_1)$
q^m_r	Lowest amount of resource $r \in R$ ever requested in a VNR
q^M_r	Highest amount of resource $r \in R$ ever requested in a VNR
w_{rv}	Weight of resource $r \in R$ required by VM $v \in N^V$
b_l	Bandwidth demand of VL $l \in L^V$
δ_l	Maximum delay allowed for VL $l \in L^V$

Table 4.1 Notation for the VNRs splitting problem

Symbols	Description
	Costs
$\mathcal{C}^N_{rai}$	Node provisioning cost of CP $i \in I$ for nodes of category $a \in A,\ r \in R$
$\mathcal{C}^L_{ti}$	Intracloud link provisioning cost of CP $i \in I$ for links of type $t \in T$
$\mathcal{C}^L_{t\varphi}$	Intercloud link provisioning cost of path $\varphi \in \mathcal{P}_{ij}$, $i, j \in I$, for links of type $t \in T$
α	Node demand weight in the total VNRs splitting cost
β	Link demand weight in the total VNRs splitting cost
	Decision variables
X_{vi}	Binary variable set to **1** if VM $v \in N^V_a$ is assigned to CP $i \in I$; **0** otherwise
$Y_{l\varphi}$	Binary variable set to **1** if VL $l \in L^V_t$ is assigned to path $\varphi \in \mathcal{P}_{ij},\ i, j \in I,\ t \in T$; **0** otherwise

4.4.2 VNR modeling

The VNR is also modeled as a weighted undirected graph $G^V = (N^V, L^V)$, where N^V is the set of VMs and L^V the set of VLs representing the inter-VMs traffic. Let N^V_a denote the set of VMs defined in category of node $a \in A$ ($\bigcup_{a \in A} N^V_a = N^V$). The required amount of resource $r \in R$ by VM $v \in N^V$ is denoted by q_{rv}. Let L^V_t designate the set of VLs of type $t \in T$ ($\bigcup_{t \in T} L^V_t = L^V$). The bandwidth demand and the maximum allowed traffic delay of VL $l \in L^V$ are respectively denoted by b_l and δ_l. This maximum delay adds constraints on the assignment of communicating VMs, in order to avoid delay violations on VLs.

4.4.3 VNRs splitting problem formulation

Our splitting strategy aims to design a VNE method with the best performance of resulting VNR segments, while satisfying resources and QoS requirements. To this end, we define the intracloud resource provisioning cost based on the availability of supplied resources. The provisioning cost of an intercloud path is stated as a function of the intracloud link provisioning cost of the two corresponding endpoints CPs and the number of PoP transit networks the path spans.

4.4.3.1 Multicloud resource provisioning costs definition

The following are the formulas that define each term and cost included in the objective function of the proposed ILP for the multicloud VNRs splitting problem.

We first define by equation (4.1) the highest available capacity of resource r supplied among all CPs, for nodes of category a, denoted by Q_{ra}^{M}. The node provisioning cost per CP is then given by equation (4.2), as the maximum available capacity of each resource the CP can offer for each category of nodes, normalized into the range [0 1] by dividing it by Q_{ra}^{M} (or by the value 1 to avoid division by zero):

$$Q_{ra}^{M} = \max_{i \in I} \left(Q_{rai} \right), \qquad \forall\ r \in R,\ a \in A \tag{4.1}$$

$$C_{rai}^{N} = \frac{Q_{rai}}{\max\left(Q_{ra}^{M}, 1\right)}, \qquad \forall\ r \in R,\ a \in A,\ i \in I \tag{4.2}$$

In order to specify the importance of each of the resources requested by VM v, equation (4.3) associates each required resource r with a weight, denoted by w_{rv}. Parameters q_r^m, and q_r^M represent respectively the lowest and highest amount of resource r ever requested by a VNR. Those values can change over the time since we consider they are estimated by the VNP based on previous statistics:

$$w_{rv} = \frac{q_{rv} - q_r^m}{q_r^M - q_r^m}, \qquad \begin{array}{l} \forall\ r \in R,\ v \in N^V, \\ q_r^m \leq q_{rv} \leq q_r^M \end{array} \tag{4.3}$$

Similarly, we define by equation (4.4) the highest available bandwidth capacity guaranteed among all CPs for each link type t, denoted by B_t^M. The intracloud link provisioning cost per CP is then given by equation (4.5), as the minimum available bandwidth capacity the CP can guarantee for each link type, normalized into the range [0 1] by dividing it by B_t^M (or by the value 1 to avoid division by zero):

$$B_t^M = \max_{i \in I} \left(B_{ti} \right), \qquad \forall\ t \in T \tag{4.4}$$

$$C_{ti}^{L} = \frac{B_{ti}}{\max(B_t^M, 1)}, \qquad \forall\ t \in T,\ i \in I \tag{4.5}$$

The intercloud link provisioning cost of each path between a pair of CPs is given by equation (4.6), as the minimum intracloud link provisioning cost between the two endpoint CPs of the path, divided by the number of PoP transit networks spanned.

$$C_{t\varphi}^{L} = \frac{\min\left(C_{ti}^{L}, C_{tj}^{L}\right)}{\max\left(\left|\mathcal{O}_{\varphi}\right|, 1\right)},\ \forall\ t \in T,\ i, j \in I, \varphi \in \mathcal{P}_{ij}. \qquad \text{with } \forall \varphi \in \mathcal{P}_{ii},\ i \in I,\ \mathcal{O}_{\varphi} = \emptyset \tag{4.6}$$

Note that if the two endpoints of a path represent the same CP (i.e. i is equal to j), equation (4.6) will be equivalent to the corresponding CP's intracloud link provisioning cost stated in equation (4.5). Since our objective function is a cost maximization, equation (4.6) will guide our solution to prefer the intracloud paths, otherwise the intercloud paths spanning the least possible PoP transit networks.

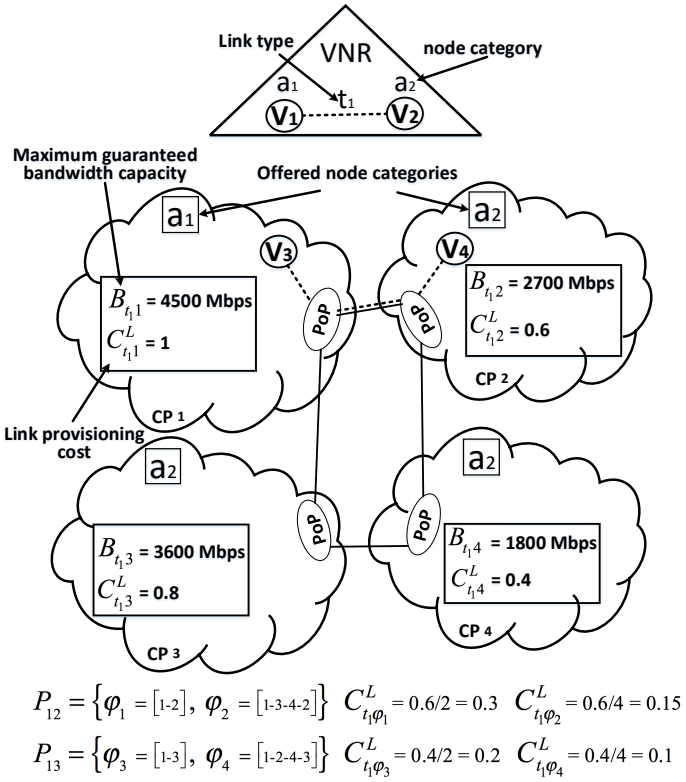

Figure 4.3 Example of a VNR splitting according to the intracloud and intercloud link provisioning cost

An example of a VNR splitting according to the intracloud and intercloud link provisioning cost is provided in Figure 4.3. Here, we suppose the four CPs having the same node provisioning cost. Thus, the request VNR_1 defined with two VMs of category a_3 exchanging traffic of type t_1, can be assigned to CP_1, CP_2 or CP_4, or split between two of the three CPs. However, the request will be exclusively assigned to CP_1 because it has the maximum intracloud link provisioning cost for links of type t_1. For the request VNR_2, it has to be split because V_3 of category a_1 can only be assigned to CP_1 and V_4 of category a_2 can be assigned to CP_2 or CP_3. In this case, V_4 will be assigned to CP_2 and the traffic will be routed on path φ_1, because the inter-domain CP_1-CP_2 has the maximum intercloud link provisioning cost and the path φ_1 spans the least PoP transit networks.

Since we use a maximization function and the number of VLs is usually much higher than the number of interconnected VMs, we need to balance the node cost and the link cost in our objective function. To this end, we define different weights for the total node demand and the total link demand of each request, respectively denoted by α and β and given by:

$$\alpha = \frac{\left|L^V\right|}{|N^V| + |L^V|}, \qquad \beta = 1 - \alpha \tag{4.7}$$

4.4.3.2 ILP formulation

The objective function of the proposed ILP model is expressed as follows:

MAX

$$\begin{aligned} &\alpha \sum_{i \in I} \sum_{r \in R} \sum_{a \in A} \sum_{v \in N_a^V} w_{rv} C_{rai}^N X_{vi} \quad + \\ &\beta \sum_{i \in I} \sum_{j \in I} \sum_{\varphi \in \mathcal{P}_{ij}} \sum_{t \in T} \sum_{l \in L_t^V} C_{t\varphi}^L Y_{l\varphi,} \end{aligned} \tag{4.8}$$

where X_{vi} is a binary variable set to 1 if VM v is assigned to CP i, 0 otherwise, and $Y_{l\varphi}$ a binary variable set to 1 if VL l is assigned to path φ, 0 otherwise. The first term of the objective function represents the total node provisioning cost for assigning VMs to CPs. The second term represents the total link provisioning cost for assigning VLs to intracloud paths (i.e where i is equal to j) or intercloud paths (i.e. where i is different from j). Note that the evaluated value in equation (4.8) is not the cost of CPs, neither the expenditure of the SP. The objective function is maximized since it represents a benefit for the SP. Indeed, the evaluated value expresses the availability of disclosed resources, and so it represents an interest for the SP (more the resources to select from a CP are available, more it is a benefit or positive interest for the SP). In contrary, if it was a cost representing the resource provisioning expenditure for the SP, the objective function would be a minimization, because the less the expenses are, more it represents a benefit for the SP.

The model is subjected to the following constraints:

$$\sum_{i \in I} X_{vi} = 1, \qquad \forall\ v \in N_a^V,\ a \in A \tag{4.9}$$

Constraint (4.9) ensures that each VM must be assigned to exactly one CP.

$$Y_{l\varphi} \leq \frac{X_{ui} + X_{vj}}{2}, \qquad \begin{array}{l} \forall\ l = uv \in L_t^V,\ t \in T, \\ u, v \in N_a^V (v \neq u),\ a \in A, \\ \varphi \in \mathcal{P}_{ij},\ i, j \in I \end{array} \tag{4.10}$$

Constraint (4.10) states the binary value of the variable $Y_{l\varphi}$ for each VL between two communicating VMs.

$$\sum_{i \in I} \sum_{j \in I} \sum_{\varphi \in \mathcal{P}_{ij}} Y_{l\varphi} = 1, \qquad \forall\ l \in L_t^V,\ t \in T \tag{4.11}$$

Constraint (4.11) ensures that each VL must be assigned to exactly one intracloud path, otherwise to an unique intercloud path.

$$\sum_{v \in N_a^V} q_{rv} X_{vi} \leq Q_{rai}, \qquad \forall\ r \in R,\ a \in A,\ i \in I \tag{4.12}$$

Constraint (4.12) ensures that the total amount of a resource required by all VMs assigned to a CP, must not exceed the maximum available capacity of the resource offered.

$$\sum_{i \in I} \sum_{j \in I} \sum_{\varphi \in \mathcal{P}_{ij}} d_\varphi Y_{l\varphi} \leq \delta_l, \qquad \forall\ l \in L_t^V,\ t \in T \tag{4.13}$$

where

$$d_\varphi = \sum_{e \in \varphi} d_e\ + \sum_{p \in O_\varphi} d_p, \qquad \forall\ \varphi \in \mathcal{P}_{ij},\ i, j \in I \tag{4.14}$$

Constraint (4.13) ensures that the total delay on a path to which a VL is assigned must be less than the maximum delay allowed for the VL.

$$X_{vi} \in \{0, 1\} \qquad \forall\ v \in N_a^V,\ a \in A,\ i \in I \tag{4.15}$$

$$Y_{l\varphi} \in \{0, 1\} \qquad \begin{array}{l} \forall\ l \in L_t^V,\ t \in T, \\ \varphi \in \mathcal{P}_{ij},\ i, j \in I \end{array} \tag{4.16}$$

Constraints (4.15) and (4.16) express the binary domain of each variable.

The proposed ILP model will allow us to provide splitting solutions that can optimize both the performance of VMs and inter-VM communication links, by selecting CPs that can offer the highest (weighted) resource capacities (node and network) at the desired locations. The

model will also minimize the number of hops (in terms of PoP transit networks spanned), and so the intercloud communication delays, while avoiding the violation of delays specified by the SP. In addition, with the defined node provisioning cost, our approach considers all the resource types that can be requested by a VM, while weighting each of the corresponding amount of demand in order to efficiently allocate VMs according to the type of application they support. For example, a VM more demanding in computation than in storage will be assigned to a provider offering much more CPU resource than disk resource (and vice-versa), which can improve the performance of the VNRs from a global point of view.

4.5 Performance evaluation

In this section, the efficiency of the proposed multicloud VNRs splitting strategy is evaluated according to several performance metrics. To this end, we compare our model to the exact solution of approaches proposed by (Mechtri *et al.*, 2017), (Leivadeas *et al.*, 2013) and (Dietrich *et al.*, 2015), which we respectively name *Distance_based*, *Scarcity_Use_based* and *LID_cost_based*. Since in (Leivadeas *et al.*, 2013) and (Dietrich *et al.*, 2015) any constraint related to substrate resources capacities and communicating delays is taken into account, two versions of our splitting model are considered in order to provide a fair comparison: 1) a "full" version, called *Full_QoS_based*, which represents the complete proposed ILP model; 2) a "simplified" version, called *Simplified_QoS_based*, where constraints (4.12) and (4.13) of the ILP formulation in Section 4.4.3.2 are removed. This will also allow us to evaluate the efficiency of the proposed splitting strategy even if such constraints are ignored.

For all *Full_QoS_based*, *Simplified_QoS_based*, *Distance_based*, *Scarcity_Use_based* and *LID_cost_based* approaches, we use the same exact solution of the adopted MILP in (Larumbe et Sanso, 2013) to perform the intracloud mapping phase presented in Appendix A.

All the ILP/MILP exact models for both the splitting and mapping phases are implemented in AMPL 64 bits with CPLEX 12.6.3 as the solver, where all scenarios are simulated on a single server with an Intel Core i7 CPU at 4 GHz and 32 GB of RAM.

In the following, we first describe the performance metrics used for the evaluation and comparison of the five approaches. Then, we present our experiments setup with different scenarios considered. Finally, we analyze and discuss obtained results.

4.5.1 Performance metrics

Here are defined the performance metrics we use for the comparative evaluation of the different approaches:

— ***Acceptance rate***: The acceptance rate represents one of the most conclusive metrics to evaluate the quality of a successful performance-based VNRs splitting strategy. The acceptance rate is measured as the number of VNRs that is successfully mapped by all CPs (including the intercloud connections), divided by the total number of VNRs.
— ***Splitting rate***: The splitting rate represents the percentage of VNRs partitions, measured as the number of assigned segments to the selected CPs at each incoming VNR, divided by the total number of CPs.
— ***Delay***: The overall delay represents the performance metric that influences the most the QoS. The delay for a packet transiting between two communicating VMs is measured as the total delay on the path between the two corresponding host DCs. If these two DCs are owned by different CPs, the total delay for the packet will accumulate the two corresponding intracloud delays, plus the intercloud delay. The overall delay is then very closely in correlation with the longer of paths and the splitting rate.
— ***Delay violations***: The delay violations are the result of an embedding that does not satisfy the delay constraints established by the SP on VLs. It is measured as the number of embedded VLs whose the resulting routing delay exceeds the maximum delay allowed on the VLs.
— ***Computing runtime***: The computing runtime represents the CPU time needed by CPLEX solver to compute the ILP/MILP models.

4.5.2 Experiments setup

We have considered two main scenarios of experiment, where VNRs arrive according to a Poisson process with an average rate of 1 request per 100 time units, with a limited lifetime uniformly distributed in the interval [1000 50000] time units:

— ***Scenario 1***, where we aim to evaluate the efficiency of the proposed splitting strategy and compare it with the other approaches in the literature based on the performance metrics listed above. In this scenario, we run 50 simulations with 5 participating CPs and 100 incoming VNRs in each simulation. The number of interconnected VMs in each VNR is randomly chosen in the interval [5, 50] (with differences of 5 VMs between requests).
— ***Scenario 2***, where the computing time complexity of our model is evaluated according to the size of the instances of the problem. The latter depends on the number of interconnected VMs and the number of participating CPs. Here, we run 10 simulations with 10 CPs and 70 incoming VNRs in each simulation. The number of interconnected VMs in each VNR is randomly chosen in the interval [2, 70] (with differences of 2 VMs between requests).

Table 4.2 Experiments parameters

	Value / distribution interval
Substrate network	
Number of nodes per CP	25
Number of links per CP	50 on average
Degree of nodes interconnectivity per CP	[3, 6]
DC CPU capacity per node category	[100, 500] cores
DC memory capacity per node category	[1000, 5000] GB
DC disk capacity per node category	[10000, 50000] GB
Intracloud link bandwidth capacity per link type	[4000, 5000] Mbps
Intracloud link delay bound	[0, 3] ms
Intercloud link / PoP transit delay bound	[0, 25] ms
Virtual network request	
VM CPU demand	[1, 40] cores
VM memory demand	[1, 400] GB
VM disk demand	[1, 4000] GB
VL bandwidth demand	[1, 40] Mbps
VL maximum delay allowed	[0, 200] ms
VNR average arrival rate	1 per 100 time units
VNR lifetime	[1000 50000] time units

All topologies and features of the substrate network and the VNRs are generated randomly using a MATLAB program and the parameters set in Table 4.2. VNRs topologies are randomly generated in a partial mesh, with 50% probability of interconnection between VMs. The multicloud substrate network interconnecting all participating CPs is generated based on ISPs topologies (Bhamare *et al.*, 2015; Doverspike *et al.*, 2010). CPs are interconnected in a half mesh connectivity. DCs in each substrate intracloud are interconnected through a backbone network, similar to the NSFNet topology (Amokrane *et al.*, 2015). Substrate nodes are located randomly at different geographic areas and each DC is connected to the backbone network through the closest routers to its location. The number of substrate nodes in each CP is set to 25, with 20% probability of generating DC nodes and 80% for router nodes, the latter being only used for forwarding purposes. The mean number of intracloud links in each CP is around 50, with a degree of node connectivity between 3 and 6. We consider that DCs are heterogeneous (can support different node categories with variable resource capacities). Three types of computational resources are considered in the scenarios, which are CPU, memory and disk. We estimated the average delay bound on each intracloud link, intercloud link and transit network as the known propagation delay, which is in the interval [0, 25] ms for intercloud links and transit networks, and in the interval [0, 3] ms for

intracloud links. Residual resource capacities for substrate nodes and links, as well as the resources advertised in the discovery framework, are dynamically updated by CPs after a VNR has been mapped or an existing VNR has been released.

Details related to the setup of cost parameters used to perform the mapping phase can be found in Appendix A.

4.5.3 Results analysis

The results analysis is first given with *Scenario 1*, by comparing *Full_QoS_based*, *Simplified_QoS_based*, *Distance_based*, *Scarcity_Use_based* and *LID_cost_based* approaches under the performance metrics described in Section 4.5.1. Thereafter, results with *Scenario 2* are presented, showing the computing time complexity of the splitting phase for the five models. For all results, the size of requests is represented in number of interconnected VMs.

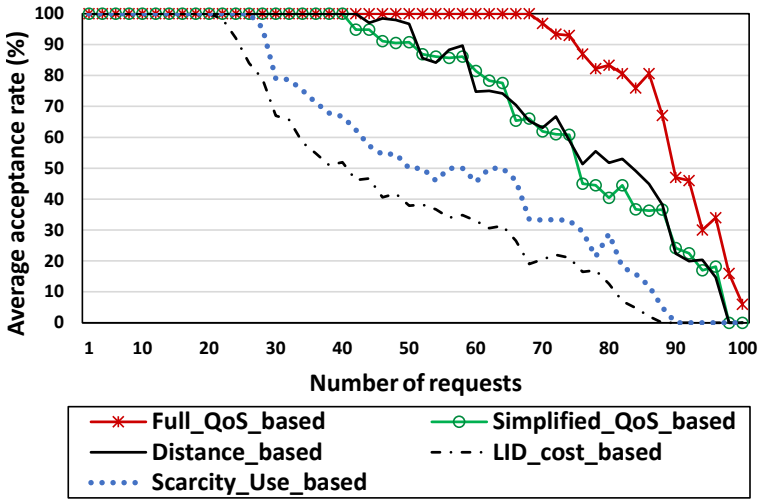

Figure 4.4 Average acceptance rate with Scenario 1. according to the VNRs arrival

4.5.3.1 Scenario 1: Comparative study of the proposed approach

First, we compare the average acceptance rate of VNRs. In our experiment, the acceptance rate for a VNR is calculated by considering both the number of VMs and the number of VLs successfully mapped. If one VM or one VL of a request has a failed embedding, we consider the entire VNR rejected. As shown in Figure 4.4, the acceptance rate decreases in average according to the arrival of requests. This is mostly due to the fact that available resources of CPs can be limited over time because of requests that have not yet expired.

Our splitting strategy, with the "full" version, leads to higher acceptance rates than the approach in (Mechtri *et al.*, 2017), (Dietrich *et al.*, 2015) and (Leivadeas *et al.*, 2013). Indeed, for *Full_QoS_based* the acceptance rate is 100% for all the first 70 VNRs, while for *Simplified_QoS_based* it is 100% for only the first 40 VNRs. *Distance_based* results in a satisfactory acceptance rate (on average better than our simplified version). However, with *Scarcity_Use_based* and more importantly with *LID_cost_based*, it decreases significantly (only 100% respectively for the first 28 and 24 VNRs) and most of the last 10 requests have a total rejection. The substantive difference between the "full" version and the "simplified" version is that the latter, by evaluating the node provisioning cost defined in equation (4.2) for each resource of a certain category of nodes, can assign VMs to the CP with the best node cost, but may at the same time violate its maximum resource capacities, unlike the "full" version. Such a situation in most cases generate a rejection of the entire request.

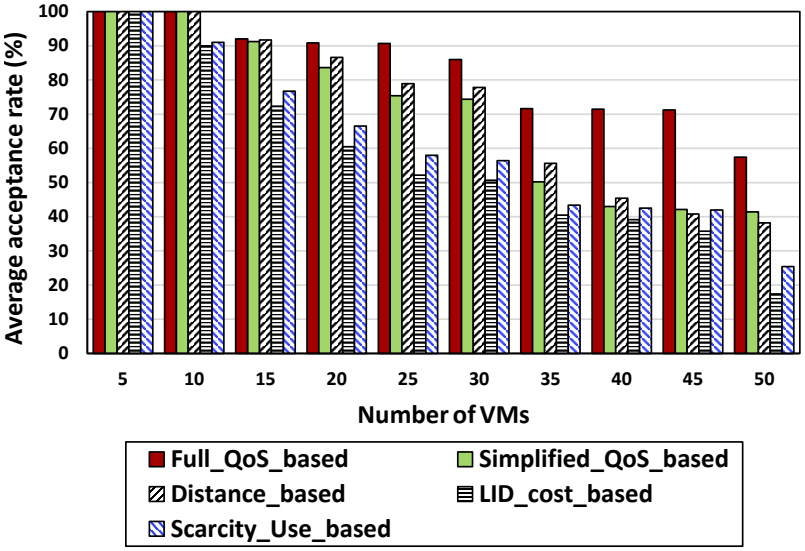

Figure 4.5 Average acceptance rate per size of request with Scenario 1.

Figure 4.5 shows the average acceptance rate per size of request, which decreases on average with the number of communicating VMs (generally the larger the VNR is, the harder it is to meet all requirements). Results demonstrate the efficiency of our splitting framework, even with the "simplified" version. We can evaluate the acceptance rate gap (percentage of acceptance rate improvement) between our model and the other comparison approaches as follows:

$$\text{Accept}_{gap} = 100 * \frac{\text{Accept}_{QoS} - \text{Accept}_{X}}{\text{Accept}_{X}}, \tag{4.17}$$

where Accept_{QoS} represents the mean acceptance rate obtained with all requests with our model, which is 87.5% for *Full_QoS_based* and 75.1% for *Simplified_QoS_based*, and Accept_{X}

represents the one obtained with the other approaches, which is 76.02% for *Distance_based*, 65.3% for *Scarcity_Use_based* and 61.2% for *LID_cost_based*. This means that with *Full_QoS_based* we improve the acceptance rate by approximately 15.1%, 34% and 43% compared to respectively *Distance_based*, *Scarcity_Use_based* and *LID_cost_based* approaches, while with *Simplified_QoS_based* we improve it only according to *Scarcity_Use_based* and *LID_cost_based* approaches, respectively by about 15% and 22.7%.

These results are due to the fact that *Distance_based*, like *Full_QoS_based*, takes into account the resource capacity constraints by evaluating a binary distance metric that eliminates unfeasible assignments, which improves on average the acceptance rate better than *Scarcity_Use_based* and *LID_cost_based*. However, the amount of requested resources are not weighted and the bandwidth capacity is not evaluated in their link provisioning cost, which makes the approach in (Mechtri *et al.*, 2017) a little less efficient than our solution. As for *LID_cost_based*, it gives the lowest acceptance rates since it is totally based on the minimization of resource prices, with no resource capacity or performance consideration. In that way, this approach will always select on average the same CPs with the same lowest resource prices (since in addition there is no dynamic pricing models implemented), and so can easily exceed CP capacities. *Scarcity_Use_based* approach gives better results than *LID_cost_based* since it evaluates the node and link performance, but tends to load most of the resource demands to CPs having the largest number of DCs and substrate links. This may not be a representative performance criterion if DCs are heterogeneous with variable capacities. Also, the bandwidth capacities on substrate paths are not necessarily related to the number of substrate links in the network.

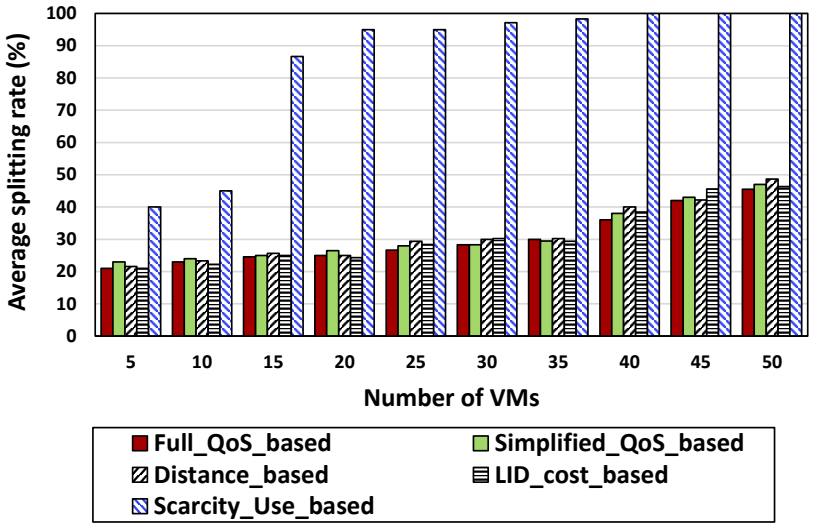

Figure 4.6 Average splitting rate with Scenario 1.

Next, we present the average VNRs splitting rate for each approach. As shown in Figure 4.6, the splitting rate increases according to the size of requests. Both versions of our splitting strategy, as well as the approaches in (Mechtri *et al.*, 2017) and (Dietrich *et al.*, 2015), give substantially same results, with an average splitting rate much lower than *Scarcity_Use_based* approach. With *Full_QoS_based* in particular, the splitting rate is on average about 21% up to 45.5%. This means that our model efficiently assigns most of the VNRs to 2 or up to 3 CPs among the 5 participants, thus avoiding useless intercloud traffics. However, *Scarcity_Use_based* approach leads to a high splitting rate, with a mean of about 96.5% considering all requests. This means that *Scarcity_Use_based* assigns most of the VNRs on average to 4 or 5 CPs, without giving better acceptance rates. A VNR splitting rate unnecessarily high should be avoidable. It generally introduces a large number of VLs transiting through the intercloud paths, and then can result in high overall intercloud delays (as shown in Figure 4.8) and also huge additional expenditure for the SP.

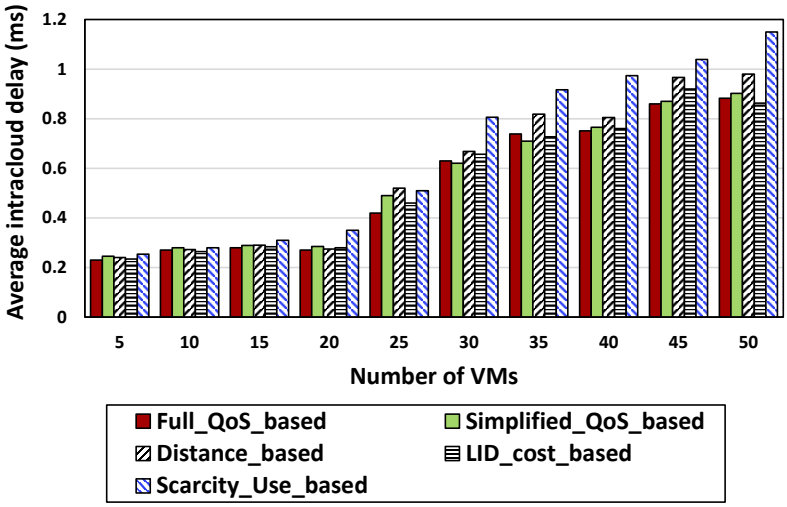

Figure 4.7 Average intracloud delay with Scenario 1.

Figure 4.7 and Figure 4.8 depict respectively the average intracloud and intercloud delays for each size of incoming requests. As expected, for all five models, the intercloud delay increases according to the size of requests and is in addition several orders of magnitude higher than the intracloud delay which is almost negligible. The "full" version of our model, considering all requests, gives on average the lowest intercloud delays, with a mean of about 7.9 ms. *LID_cost_based* and *Distance_based* generate also low intercloud delays, respectively with a mean of about 9.7 ms and 9.85 ms, compared to our "simplified" version and the approach in (Leivadeas *et al.*, 2013) resulting respectively with a mean of about 11.8 ms and 23.2 ms. *Distance_based* and *LID_cost_based* give such results because the former considers the

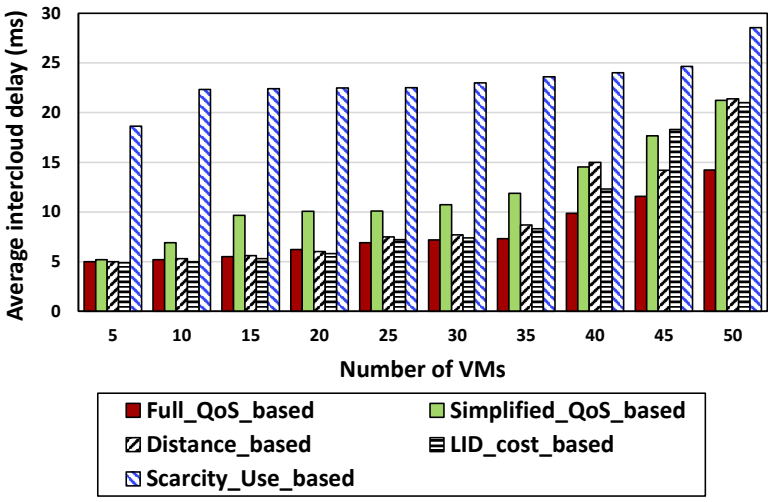

Figure 4.8 Average intercloud delay with Scenario 1.

communication delay which is reduced, and the latter minimizes the accumulated link cost which is closely related to the length of paths and the number of PoPs spanned. As a result, the intercloud delay has on average a better improvement than with *Simplified_QoS_based* and *Scarcity_Use_based*. *Simplified_QoS_based*, by ignoring the delay sensitivity of applications, provides on average solutions with higher splitting rates, and so interCloud delays, than *Full_QoS_based*.

By evaluating the intercloud delay gap with the same formula as in (4.17), we can notice that our "full" approach improves the intercloud delay by about 18.5% and 19.7% compared to respectively *LID_cost_based* and *Distance_based*, while with *Scarcity_Use_based* it is improved by about 66%. We can also observe that the splitting rate has a significant influence on the intercloud delay, by resulting in more VMs communicating through intercloud VLs. Thus, our approach penalizes more effectively long intercloud paths than the other approaches.

We also present the average delay violations on VLs. As shown in Figure 4.9, although any delay constraint is considered with *Simplified_QoS_based* and *LID_cost_based*, the latter are much less subject to delay violations than *Scarcity_Use_based*. *Full_QoS_based* and *Distance_based* are nevertheless subject to some delay violations, even though the delay constraint is taken into account. Indeed, as explained above, solutions with the multicloud VNE problem are not always optimal from the global view of the embedding process due to the non-interoperability between CPs. However, we can see that globally, our splitting strategy, even with the "simplified" version, leads to greater QoS satisfactions by minimizing communication delays between VMs, while preventing delay violations.

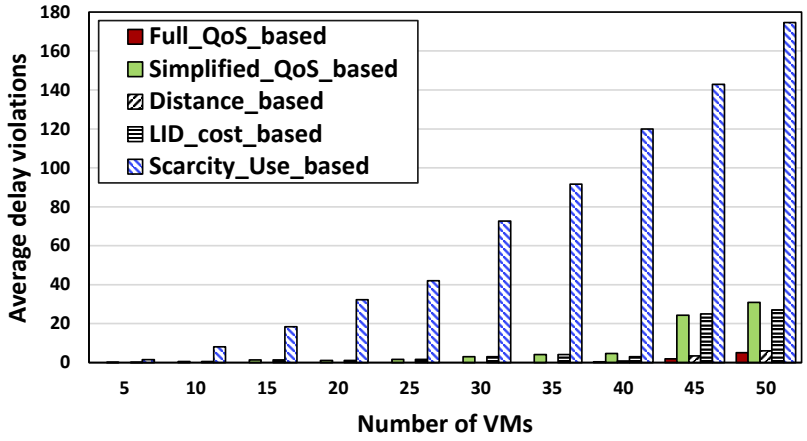

Figure 4.9 Average delay violations with Scenario 1.

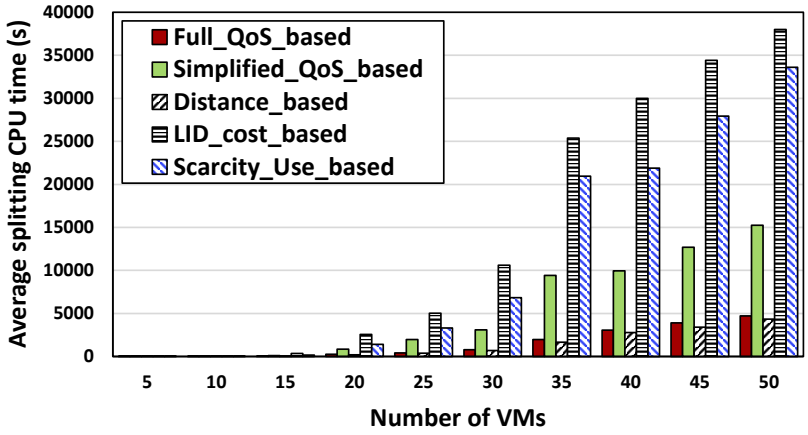

Figure 4.10 Average processing time for the splitting phase with Scenario 1.

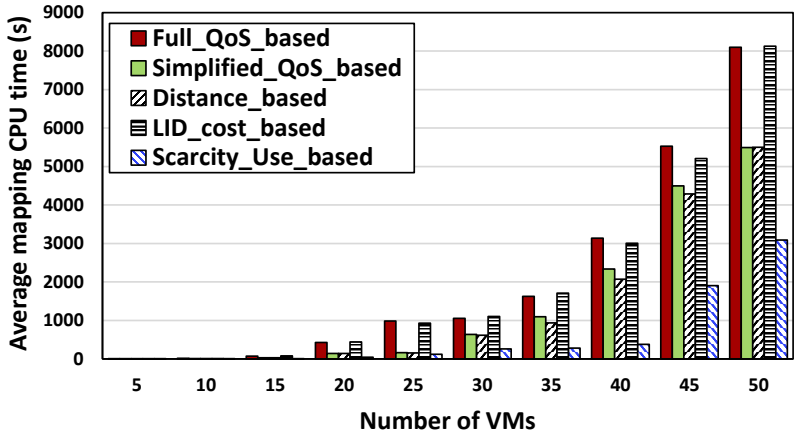

Figure 4.11 Average processing time for the mapping phase with Scenario 1.

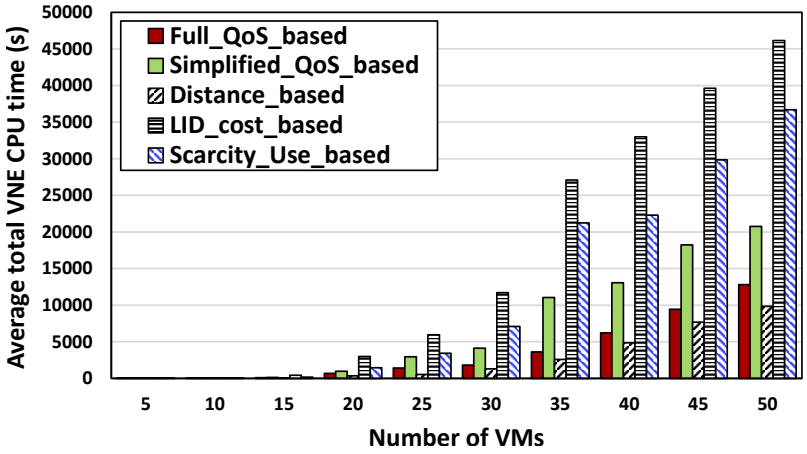

Figure 4.12 Total average processing time of VNE with Scenario 1.

Finally, we present in Figure 4.10, Figure 4.11 and Figure 4.12 the average CPU time needed to compute the splitting and mapping phases for the five approaches. The execution time is on average in correlation with the number of interconnected VMs (and thus with the number of VLs). For both phases it increases according to the size of requests. As shown in Figure 4.10, *Full_QoS_based* and *Distance_based* models take less time to execute the splitting phase, even with large sized requests with 45 or 50 VMs. This is due to the additional constraints the two models take into account, which gives a more reduced search space for the solver. Indeed, *Simplified_QoS_based*, *Scarcity_Use_based* and *LID_cost_based* considering less constraints, take more time to compute all possible solutions to find the optimum. As for *LID_cost_based*, it generates the highest execution times for the splitting phase because the VMs assignment is evaluated with the peering nodes instead of the CPs, which makes the complexity of the problem higher as the number of peering nodes are larger than the number of CPs. On the other hand, as shown in Figure 4.11, *Full_QoS_based* and *LID_cost_based* generate the highest processing times to perform the mapping phase. *Scarcity_Use_based* gives the lowest mapping processing times, compared to *Simplified_QoS_based* and *Distance_based* for which it is similarly higher than for *Scarcity_Use_based*.

These results are explained with the splitting rates presented in Figure 4.6. Indeed, *Scarcity_Use_based* approach giving the highest splitting rates, generates then many small VNR segments that are easier to compute at the mapping phase, compared to *Full_QoS_based* and *LID_cost_based*, for example, which generate fewer VNR segments but on average with larger sizes requiring more mapping processing time. Figure 4.12 shows the total processing time to split and map the requests. From a global point of view, *Distance_based* takes the least time to perform the whole embedding process, followed by our approach in the two

versions. As expected, *LID_cost_based* take the highest time for the entire VNE process, compared to *Scarcity_Use_based*.

4.5.3.2 Scenario 2: Analysis of the computing time complexity of the splitting phase

The second scenario aims at analyzing the computing time complexity for the five models with larger instances of the splitting problem. Table 4.3 shows the results of this analysis. For each of the five models, the mean CPU time is given for each size of request. We arbitrary set a time limit (TL) of splitting processing to 21,600 s and the proportion of request instances reaching the TL is also indicated in the table. As expected, the processing time taken by the solver increases exponentially in correlation with the number of interconnected VMs. In fact, the size of requests is more significantly related to the number of VLs. Indeed, as VMs are interconnected in partial mesh, with n VMs, the number of VLs (in both directions of communication) increases considerably in the range of $n\,(n-1)/2$. As CPLEX solver uses Branch-and-Bound methods to compute the ILP model, the branch-and-cut tree increases drastically with the large number of VLs that have to be embedded and all possible paths between each pair of the 10 CPs to consider. This leads the processing time to increase exponentially.

The processing time complexity is also related to the search space of solutions. Indeed, we can notice from Table 4.3 that, with *Full_QoS_based* and *Distance_based* approaches having more reduced search space, the solver can process requests respectively with up to 54 VMs and 56 VMs, before being limited by time. For *Full_QoS_based*, the TL has mostly been reached by requests with 60 VMs (about 77.6%), and by all requests with more than 62 VMs, without the proof of optimality of the obtained solution being established. *Distance_based* is a little faster, with the TL mostly reached by requests with 62 VMs (about 87.3%), and by all requests with more than 64 VMs. However, unsurprisingly, *Simplified_QoS_based*, *Scarcity_Use_based* and *LID_cost_based* models converge more quickly towards the TL. Indeed, the TL has often been reached by requests with 30 VMs (about 89.7%) for *Simplified_QoS_based*, 24 VMs (about 78.7%) for *Scarcity_Use_based* and only 22 VMs (about 22.4%) for *LID_cost_based*, and by all requests respectively with more than 32 VMs, 26 VMs and 24 VMs.

Such results with the scenario 2 show that it is hard to perform, in a reasonable execution time, the VNE on large-scale instances of the problem with exact approaches. In such context, it would be necessary to have a very powerful computing device to run the ILP models, otherwise to perform a heuristic approach.

Table 4.3 Computing time complexity of the splitting phase with Scenario 2

	Full QoS based		*Simplified QoS based*		*Distance based*		*LID cost based*		*Scarcity Used based*	
Number of VMs	Mean CPU time (s)	% TL reached	Mean CPU time (s)	% TL reached	Mean CPU time (s)	% TL reached	Mean CPU time (s)	% TL reached	Mean CPU time (s)	% TL reached
2	0.07	0	0.24	0	0.07	0	0.22	0	0.19	0
4	2.02	0	1.9	0	1.4	0	3.1	0	0.6	0
6	2.8	0	3.7	0	2.4	0	5.6	0	3.3	0
8	19.1	0	26	0	18.3	0	41.4	0	45.2	0
10	88.7	0	159.7	0	85.7	0	168.3	0	137.1	0
12	90.12	0	271.6	0	92.5	0	490.4	0	361.5	0
14	136.3	0	1028.4	0	130.8	0	1412.2	0	1167.8	0
16	371.7	0	1797.6	0	354.9	0	2278.3	0	1988.7	0
18	621.8	0	3306.8	0	547	0	3988.2	0	3397.3	0
20	730.7	0	5252.7	0	615.4	0	6346.9	0	5724.4	0
22	971.3	0	5735.4	0	846.1	0	10495.5	22.4	8749.9	0
24	1068.2	0	10851	0	955.3	0	TL	100	TL	78.7
26	1274.2	0	15185.5	0	1002.3	0	TL	100	TL	100
28	1695.2	0	20743.7	46.3	1238.9	0	TL	100	TL	100
30	1868.2	0	TL	89.7	1724.8	0	TL	100	TL	100
32	1945.4	0	TL	100	1701.6	0	TL	100	TL	100
34	2861.1	0	TL	100	2637	0	TL	100	TL	100
36	3739.6	0	TL	100	3498.3	0	TL	100	TL	100
38	4534.4	0	TL	100	4615.7	0	TL	100	TL	100
40	5142.2	0	TL	100	4893.3	0	TL	100	TL	100
42	5628.9	0	TL	100	5277.6	0	TL	100	TL	100
44	7593.1	0	TL	100	7402.3	0	TL	100	TL	100
46	8534.4	0	TL	100	8290	0	TL	100	TL	100
48	10155.8	0	TL	100	9000.4	0	TL	100	TL	100
50	11799.8	0	TL	100	11025.2	0	TL	100	TL	100
52	14935.2	0	TL	100	14501.7	0	TL	100	TL	100
54	17257.2	0	TL	100	17326.6	0	TL	100	TL	100
56	19723.2	23.1	TL	100	18994.8	0	TL	100	TL	100
58	19978.3	38.4	TL	100	19504	10.8	TL	100	TL	100
60	20049.1	77.6	TL	100	19984.3	42.5	TL	100	TL	100
62	TL	100	TL	100	TL	87.3	TL	100	TL	100
64	TL	100	TL	100	TL	100	TL	100	TL	100
66	TL	100	TL	100	TL	100	TL	100	TL	100
68	TL	100	TL	100	TL	100	TL	100	TL	100
70	TL	100	TL	100	TL	100	TL	100	TL	100

4.6 Conclusion

In this paper, the problem of VNE across multiple IaaS-based cloud providers has been addressed. A multicloud VNRs splitting strategy has been proposed with the objective of improving the performance and the QoS of resulting VNR segments that are mapped onto the selected cloud infrastructure networks. A mathematical ILP model has been used to formalize the proposed VNRs splitting strategy as a maximization problem with constraints. Thereafter, a multi-criteria cost-minimization solution, formalized as a MILP, has been adopted in order to reduce mainly the overall delay during the intracloud mapping phase. Our approach was compared to other baselines approaches through simulations. Numerical results have shown the efficiency of the proposed strategy to deal with the scalability of the problem. Our strategy has improved several performance criteria, including the acceptance rate and the delay, while providing a VNRs embedding solution that best satisfies QoS requirements.

Since the VNE problem is classified as an NP-hard problem, it is worthwhile to address in a future work the development of faster large-scale heuristic algorithms for each of the splitting and mapping phases. Furthermore, the proposed mathematical formulation can be extended to a multi-objective approach, by also considering the expenditure minimization of the SP in the resource provisioning during the splitting phase.

Appendix A FORMULATION OF THE VNR SEGMENTS INTRACLOUD MAPPING PROBLEM

In this appendix, we first give a brief description of the CP's intracloud network and the VNR segment modelings. Then, we present the MILP formulation of the adopted intracloud mapping approach proposed in (Larumbe et Sanso, 2013). All the notation used can be consulted in Table 4.4. Details related to the setup of cost parameters used for our simulations are also given.

Table 4.4 Notation for the VNR segments intracloud mapping problem

Symbols	Description
	Global sets
A	Set of node categories
T	Set of link types
R	Set of computational resource types
I	Set of CPs
	Substrate intracloud network
G_i^S	Graph representing the substrate network of CP $i \in I$
N_i^S	Set of substrate nodes in G_i^S
L_i^S	Set of substrate links in G_i^S
D_i	Set of DCs of CP $i \in I$
ϕ_i	PoP node (transit network) of CP $i \in I$
$\mathcal{F}_i$	Set of all paths in CP i's network
$\mathcal{P}_{dc}$	Set of all paths between DCs d and c, $d, c \in D_i$
$\mathcal{P}_{d\phi_i}$	Set of all paths between DC $d \in D_i$ and PoP node ϕ_i
$\mathcal{K}_e$	Set of all paths in $\mathcal{F}_i$ spanning link $e \in L_i^S$
B_{te}	Available bandwidth capacity of the channel of type $t \in T$ of link $e \in L_i^S$
d_e	Delay bound on link $e \in L_i^S$
Q_{rad}	Available capacity of resource $r \in R$ in DC $d \in D_i$for nodes of category $a \in A$ ($Q_{rad} \in \mathbb{N}_1$)
U_{adr}	Usage of resource $r \in R$ by all VMs of node category $a \in A$ assigned to DC $d \in D_i$
E_{adr}	Average power (in watts) consumed by a VM of node category $a \in A$ assigned to DC $d \in D_i$ in terms of resource $r \in R$
ω_d	Average power (in watts) consumed by all VMs assigned to DC $d \in D_i$
ρ_d	Power Usage Effectiveness (PUE) of DC $d \in D_i$
θ_d	CO_2 emissions in DC $d \in D_i$ (in g/KWh)
	Virtual network request segment
G_i^V	Graph representing the VNR segment assigned to CP $i \in I$
N_i^V	Set of VMs assigned to CP $i \in I$
L_i^V	Set of virtual links assigned to CP $i \in I$
N_{ai}^V	Set of VMs of node category $a \in A$ assigned to CP $i \in I$
L_{ti}^V	Set of virtual links of type $t \in T$ exclusively assigned to CP $i \in I$
$L_{t\phi_i}^V$	Set of virtual links of type $t \in T$ assigned to an inter-cloud link of endpoint CP $i \in I$
q_{rv}	Amount of resource $r \in R$ required by VM $v \in N_i^V$
b_l	Bandwidth demand of virtual link $l \in L_i^V$
δ_l	Maximum delay allowed for virtual link $l \in L_i^V$

Table 4.4 Notation for the VNR segments intracloud mapping problem

	Costs
$c^{\mathcal{S}}_{rai}$	Unit resource cost for CP $i \in I$ for using resource $r \in R$ for nodes of category $a \in A$ (in \$/unit)
c_{ti}	Unit bandwidth cost for CP $i \in I$ for using a link channel of type $t \in T$ (in \$/Mbps)
$c^{\mathcal{E}}_{ti}$	Extra unit bandwidth cost for CP $i \in I$ for using a link channel of type $t \in T$ (in \$/Mbps)
c^{ω}_{d}	Unit electricity cost in DC $d \in D_i$ (in \$/MWh)
$\wp^{\mathcal{D}}$	Penalty for each millisecond of delay on virtual links (in \$/ms)
$\wp^{\mathcal{O}}_{d}$	Penalty for emitting CO_2 in DC $d \in D_i$ (in \$/tonne)
$C^{\mathcal{S}}_{i}$	Total computing resource cost for CP $i \in I$ (in \$/h)
$C^{\mathcal{T}}_{i}$	Total traffic cost for CP $i \in I$ (in \$/h)
$C^{\mathcal{D}}_{i}$	Total delay penalty for CP $i \in I$ (in \$/h)
C^{ω}_{i}	Total energy cost for CP $i \in I$ (in \$/h)
$C^{\mathcal{O}}_{i}$	Total environmental penalty for CP $i \in I$ (in \$/h)
	Decision variables
X_{vn}	Binary variable set to **1** if VM $v \in N^V_{ai},\ a \in A$, is assigned to substrate node $n \in N^S_i$; **0** otherwise
$Y_{l\varphi}$	Binary variable set to **1** if virtual link $l \in L^V_{ti},\ t \in T$, is assigned to path $\varphi \in \mathcal{P}_{dc}$, $d, c \in D_i$; **0** otherwise
$Z_{k\gamma}$	Binary variable set to **1** if virtual link $k \in L^V_{t\phi_i},\ t \in T$, is assigned to path $\gamma \in \mathcal{P}_{d\phi_i}$, $d \in D_i$; **0** otherwise

A.1 Intracloud network and VNR segment modelings

The substrate network of each CP $i \in I$ (set of CPs) is modeled as a weighted undirected graph $G^S_i = (N^S_i,\ L^S_i)$, where N^S_i is the set of substrate nodes and L^S_i the set of substrate links. Substrate nodes in N^S_i include DCs, denoted by D_i, routers and peering nodes. The PoP network of each CP i is represented by a single node denoted by ϕ_i, since we do not manage routing inside the PoP network. Routers are only used for forwarding purposes. We assume that DCs are heterogenous, which means that they have not the same capacities and a single DC may support many node categories in set A. Moreover, each DC is associated with each computational resource type in set R. Let Q_{rad} denote the residual capacity of resource $r \in R$ in DC $d \in D_i$ for node category $a \in A$. Every substrate link $e \in L^S_i$ supports a channel for each link type in set T, with a delay bound denoted by d_e. Each channel type is associated with a residual bandwidth capacity denoted by B_{et}. Let $\mathcal{F}_i$ denote the set of all paths in CP i's network. The set of paths between each pair of DCs d and c is denoted by $\mathcal{P}_{dc}$. The set of all paths between DC d and the PoP node ϕ_i is denoted by $\mathcal{P}_{d\phi_i}$. Finally, let $\mathcal{K}_e (\subseteq \mathcal{F}_i)$ denote the set of all paths spanning link $e \in L^S_i$.

The VNR segment is also modeled as a weighted undirected graph $G^V_i = (N^V_i,\ L^V_i)$, where N^V_i and L^V_i represent respectively the set of VMs and the set of virtual links assigned to CP

$i \in I$. We denote by N_{ai}^{V} the set of VMs of node category $a \in A$ assigned to CP i. The set of virtual links of type $t \in T$ exclusively assigned to CP i is denoted by L_{ti}^{V}, while the set of virtual links of type $t \in T$ assigned to an inter-cloud link of endpoint CP i is denoted by $L_{t\phi_i}^{V}$. VMs are mapped into the DCs, as well as virtual links that interconnect VMs hosted in the same DC. Any virtual link in set $L_{t\phi_i}^{V}$ is assigned to a path in set $\mathcal{P}_{d\phi_i}$.

A.2 Adopted intracloud mapping approach

The following are the formulas that define each cost and penalty of the multicriteria objective function. In our solution, the mapping of a VM remains at the DC level. VMs requirements are expressed in terms of computational resource demand in set R, not in terms of number of servers. Therefore, some formulas originally defined in (Larumbe et Sanso, 2013) were re-stated. Moreover, we did not consider the path splitting scenario, neither the DCs CAPEX and OPEX since we assign VMs to existing DCs. Note that each cost is defined in dollars per hour ($/h).

Servers cost is re-stated as computational resources cost and is calculated as follows:

$$C_i^{\mathcal{S}} = \sum_{d \in D_i} \sum_{r \in R} \sum_{a \in A} \sum_{v \in N_{ai}^{V}} c_{rai}^{\mathcal{S}} q_{rv} X_{vd} \tag{A.1}$$

The total traffic cost for CP i is defined as follows, with an extra cost stated for inter-cloud virtual links:

$$\begin{aligned} C_i^{\mathcal{T}} = & \sum_{d \in D_i} \sum_{c \in D_i} \sum_{\varphi \in \mathcal{P}_{dc}} \sum_{e \in \varphi} \sum_{t \in T} \sum_{l \in L_{ti}^{V}} c_{ti} b_l Y_{l\varphi} \\ & + \sum_{d \in D_i} \sum_{\gamma \in \mathcal{P}_{d\phi_i}} \sum_{e \in \gamma} \sum_{t \in T} \sum_{k \in L_{t\phi_i}^{V}} \left(c_{ti} + c_{ti}^{\mathcal{E}} \right) b_k Z_{k\gamma} \end{aligned} \tag{A.2}$$

The total delay penalty for CP i is re-defined as follows:

$$C_i^{\mathcal{D}} = \wp^{\mathcal{D}} \left(\begin{array}{l} \sum_{d \in D_i} \sum_{c \in D_i} \sum_{\varphi \in \mathcal{P}_{dc}} \sum_{e \in \varphi} \sum_{t \in T} \sum_{l \in L_{ti}^{V}} d_e Y_{l\varphi} \\ + \sum_{d \in D_i} \sum_{\gamma \in \mathcal{P}_{d\phi_i}} \sum_{e \in \gamma} \sum_{t \in T} \sum_{k \in L_{t\phi_i}^{V}} d_e Z_{k\gamma} \end{array} \right) \tag{A.3}$$

The average power consumed by all VMs assigned to DC $d \in D_i$ is re-stated as follows, estimated as a function of the utilization of CPU resource and disk resource, by following the principles used in (Justafort *et al.*, 2015):

$$\omega_d = \sum_{a \in A} \left(E_{adr_1} U_{adr_1} + E_{adr_2} U_{adr_2} \right), \quad \forall\, d \in D_i, \tag{A.4}$$

where E_{adr_1} and E_{adr_2} represent (in watts) the average power consumed by a VM of node category $a \in A$ assigned to DC d, respectively in terms of CPU resource and disk resource. U_{adr_1} and U_{adr_2} define respectively the CPU resource usage and the disk resource usage by all VMs of node category $a \in A$ assigned to DC d, given by:

$$U_{adr} = \frac{\sum\limits_{v \in N_{ai}^V} q_{rv} X_{vd}}{\max\left(Q_{rad}, 1\right)}, \qquad \forall\, d \in D_i,\ a \in A,\ r \in R \tag{A.5}$$

The total energy cost for CP i is then defined as follows:

$$C_i^{\omega} = 10^{-6} \sum_{d \in D_i} c_d^{\omega} \rho_d \omega_d, \tag{A.6}$$

The total environmental penalty for CP i is given as follows:

$$C_i^{\mathcal{O}} = 10^{-9} \sum_{d \in D_i} \wp_d^{\mathcal{O}} \theta_d \rho_d \omega_d \tag{A.7}$$

The objective function for the intracloud mapping phase is then defined as follows, with each cost weighted by a parameter that allows CPs to modify the costs priority:

MIN

$$\alpha \mathcal{C}_i^{\mathcal{S}} + \beta \mathcal{C}_i^{\mathcal{T}} + \lambda \mathcal{C}_i^{\mathcal{D}} + \varpi \mathcal{C}_i^{\omega} + o \mathcal{C}_i^{\mathcal{O}} \tag{A.8}$$

Subject to:

$$\sum_{n \in N_i^S \setminus D_i} X_{vn} = 0, \qquad \forall\, v \in N_{ai}^V,\ a \in A \tag{A.9}$$

$$\sum_{d \in D_i} X_{vd} = 1, \qquad \forall\, v \in N_{ai}^V,\ a \in A \tag{A.10}$$

Constraints (A.9) and (A.10) ensure respectively that VMs are only assigned to DCs, and each VM must be assigned to exactly one DC.

$$Y_{l\varphi} \leq \frac{X_{ud} + X_{vc}}{2}, \quad \begin{array}{l} \forall\, l = uv \in L_{ti}^V,\ t \in T, \quad u, v \in N_{ai}^V (v \neq u),\ a \in A, \\ \varphi \in \mathcal{P}_{dc},\ d, c \in D_i \end{array} \tag{A.11}$$

$$Z_{k\gamma} \leq X_{vd}, \qquad \forall\, k = v\phi_i \in L_{t\phi_i}^V,\ t \in T,\ v \in N_{ai}^V, \quad \gamma \in \mathcal{P}_{d\phi_i},\ d \in D_i \tag{A.12}$$

Constraints (A.11) and (A.12) state respectively the binary value of variables $Y_{l\varphi}$ and $Z_{k\gamma}$.

$$\sum_{d\in D_i}\sum_{c\in D_i}\sum_{\varphi\in\mathcal{P}_{dc}} Y_{l\varphi}=1, \qquad \forall\ l\in L_{ti}^{V},\ t\in T \tag{A.13}$$

$$\sum_{d\in D_i}\sum_{\gamma\in\mathcal{P}_{d\phi_i}} Z_{k\gamma}=1, \qquad \forall\ k\in L_{t\phi_i}^{V},\ t\in T \tag{A.14}$$

Constraint (A.13) ensures that each virtual link exclusively assigned to a CP must be mapped to a unique path between two different DCs, otherwise to a unique path intra-DC. Constraint (A.14) ensures that each virtual link partially assigned to a CP must be mapped to a unique path between a DC and the PoP node of the CP.

$$\sum_{v\in N_{ai}^{V}} q_{rv}X_{vd}\leq Q_{rad}, \qquad \forall\ r\in R,\ a\in A,\ d\in D_i \tag{A.15}$$

$$Q_{rad}=Q_{rad}-\sum_{v\in N_{ai}^{V}} q_{rv}X_{vd}, \qquad \forall\ r\in R,\ a\in A,\ \ d\in D_i \tag{A.16}$$

Constraint (A.15) ensures that the total amount of a resource required by all VMs assigned to a DC must not exceed its residual capacity. Constraint (A.16) updates this residual capacity after each VNR is successfully mapped.

$$B_{\overrightarrow{te}}=B_{\overleftarrow{te}}, \qquad \forall\ e\in L_{ti}^{S} \tag{A.17}$$

Constraint (A.17) states that the bandwidth capacity of a channel of a link type is the same in both directions on every substrate link.

$$\begin{gathered}\sum_{m\in N_i^S} f_{nm}^{uv}+X_{vn}b_l=\sum_{m\in N_i^S} f_{mn}^{uv}+X_{un}b_l,\\ \forall u,v\in N_i^V,\ v\neq u, n\in N_i^S,\ l=uv\in L_i^V\end{gathered} \tag{A.18}$$

Constraint (A.18) guaranties the flow conservation for every amount of traffic $l=uv$ from VM u to VM v, with a bandwidth demand b_l routed on substrate link $e=mn\in L_i^S$.

$$\begin{gathered}\sum_{\varphi\in\mathcal{K}_e}\sum_{l\in L_{ti}^{V}} b_l Y_{l\varphi}+\sum_{\gamma\in\mathcal{K}_e}\sum_{k\in L_{t\phi_i}^{V}} b_k Z_{k\gamma}\leq B_{te},\\ \forall\ e\in L_i^S,\ t\in T\end{gathered} \tag{A.19}$$

$$\begin{gathered}B_e=B_e-\left(\sum_{\varphi\in\mathcal{K}_e}\sum_{l\in L_{it}^{V}} b_l Y_{l\varphi}+\sum_{\gamma\in\mathcal{K}_e}\sum_{k\in L_{t\phi_i}^{V}} b_k Z_{k\gamma}\right),\\ \forall\ e\in L_{ti}^{S},\ t\in T\end{gathered} \tag{A.20}$$

Constraint (A.19) ensures that the total bandwidth demand of all virtual links routed on a substrate link must not exceed the residual bandwidth capacity of the corresponding channel. Constraint (A.20) updates this residual capacity after a VNR is successfully mapped.

$$\sum_{d\in D_i}\sum_{c\in D_i}\sum_{\varphi\in\mathcal{P}_{dc}}\sum_{e\in\varphi} d_e Y_{l\varphi} \leq \delta_l,\ \forall\ l \in L_{ti}^V,\ t \in T \tag{A.21}$$

$$\sum_{d\in D_i}\sum_{\gamma\in\mathcal{P}_{d\phi_i}}\sum_{e\in\gamma} d_e Z_{l\gamma} \leq \delta_k,\ \ \forall\ k \in L_{t\phi_i}^V,\ t \in T \tag{A.22}$$

Constraints (A.21) and (A.22) ensure that the restriction on the maximum delay allowed for a virtual link is not violated.

$$X_{vn} \in \{0,1\} \qquad \forall\ v \in N_{ai}^V,\ a \in A,\ n \in N_i^S \tag{A.23}$$

$$Y_{l\varphi} \in \{0,1\} \quad \forall\ l \in L_{ti}^V,\ t \in T,\ \varphi \in \mathcal{P}_{dc},\ d, c \in D_i \tag{A.24}$$

$$Z_{k\gamma} \in \{0,1\} \quad \forall\ k \in L_{t\phi_i}^V,\ t \in T,\ \gamma \in \mathcal{P}_{d\phi_i},\ d \in D_i \tag{A.25}$$

$$\omega_d \in \mathbb{R}_{\geq 0} \qquad \forall\ d \in D_i \tag{A.26}$$

Constraints (A.23) - (A.26) express the domain of definition of each variable.

A.3 Cost parameters setup

Most of cost parameters setting follows the experimentation setup defined in (Larumbe et Sanso, 2013), but they are specified for an hour time period. We chose the delay as the first optimization priority of the multi-objective function, by using the same weight attribution as in (Larumbe et Sanso, 2013). The unit computational resource cost is randomly determined in the interval \$[1, 5]/core/h for CPU resource, \$([1, 5]/10)/GB/h for memory resource and \$([1, 5]/100)/GB/h for disk resource. The unit bandwidth cost for each link type is randomly chosen in the interval \$([5, 10]/1000)/Mbps/h, and the corresponding extra unit bandwidth cost is estimated at 25% of the initial unit cost. The delay penalty is set to \$1.15/(ms/packet)/h. The average power consumed by a VM assigned to a data center is randomly set in the interval [200 300] W (with 60% of the energy for CPU resource and 40% for disk resource). The unit electricity cost is randomly distributed in the interval \$[30 70]/MWh. The PUE of a data center is 1.5. The amount of CO_2 emissions in a data center is calculated by summing values in the set {10 66 443 960} g/KWh, each of them first multiplied by a greenness factor randomly set between 0 and 1. The penalty for emitting CO_2 in a data center is defined as \$1000/tonne.

CHAPITRE 5 ARTICLE 2 : A TABU SEARCH APPROACH FOR A VIRTUAL NETWORKS SPLITTING STRATEGY ACROSS MULTIPLE CLOUD PROVIDERS

Auteurs : Marieme Diallo, Alejandro Quintero et Samuel Pierre.
Revue : Accepté pour publication dans le journal *International Journal of Metaheuristics*, en Août 2018.

Abstract

This paper addresses the problem of computational and networking resources embedding across multiple independent cloud providers (CPs). We focus on the splitting phase problem by proposing a virtual network requests (VNRs) splitting strategy, which aims at improving the performance and the quality of service (QoS) of resulting mapped VNR segments. We formalize our splitting strategy as a mathematical maximization problem with constraints by using an Integer Linear Program (ILP). Since the VNRs splitting process is classified as an NP-hard problem, we propose a metaheuristic approach based on the Tabu Search (TS), in order to find good feasible solutions in polynomial solving time. The simulations results obtained show the efficiency of the proposed algorithm, in comparison with the exact method and an other baseline approach. Solution costs are on average close to the upper bounds, with an average gap ranging from 0% to a maximum of 2.97%, performed in a highly reduced computing time.

5.1 Introduction

Cloud computing has recently emerged as an innovative utility computing solution (Armbrust *et al.*, 2010), allowing small businesses to rent distributed configurable resources as an on-demand service model. Nowadays, the Infrastructure as a Service (IaaS) has become the most widely adopted cloud service model (Manvi et Shyam, 2014). In this paradigm, a Service Provider (SP) can lease a large pool of virtualized infrastructure layer resources (computational and networking) from one or more Cloud Providers (CPs), in order to build heterogeneous virtual networks that will offer customized services to its end clients.

In the IaaS business model, a fundamental management problem lies in the efficient embedding of co-existing virtual network requests (VNRs) onto distributed substrate infrastructures (Belbekkouche *et al.*, 2012; Manvi et Shyam, 2014). This issue, usually referred to as the

well-known NP-hard Virtual Network Embedding (VNE) problem (Chowdhury *et al.*, 2012; Fischer *et al.*, 2013; Zhang *et al.*, 2016a), becomes more challenging when the substrate infrastructures are owned by multiple independent CPs (Grozev et Buyya, 2012; Rafael *et al.*, 2012). Indeed, in such a context, the VNE process requires two major phases of operation, each of them dealing with an NP-Hard problem with different technical approaches to resolve it: the multicloud VNRs splitting phase, followed by the intracloud VNR segments mapping phase. In the first phase, which is similar to a graph partitioning problem (Sanchis, 1989; Tao *et al.*, 1992), a virtual network provider (VNP), acting as a virtual brokerage service on behalf of the SP (Fischer *et al.*, 2013), generally uses a strategy to select eligible CPs based on the SP's requirements, and split the VNRs into different segments. In the second phase, which corresponds to the VNE problem within a single CP (Chowdhury *et al.*, 2012; Khan *et al.*, 2016; Zhang *et al.*, 2014), each selected CP uses a mapping approach to embed the assigned VNR segments into its intracloud network.

VNE over a multicloud network has been only recently addressed in the literature (Dietrich *et al.*, 2015; Leivadeas *et al.*, 2013; Mano *et al.*, 2016; Mechtri *et al.*, 2017). It adds more complexity and scalability issues. Indeed, due to the non-interoperability between CPs, it becomes difficult to have optimal configurations from the global view of the multicloud embedding process. Moreover, the VNP visibility on the multicloud substrate network is essentially decisive for the efficiency of any VNRs splitting strategy. However, the VNP proceeds with the splitting phase based on a very poor knowledge of the multicloud environment. Information such as substrate network topologies and details about the resources availabilities are usually concealed by the CPs (Dietrich *et al.*, 2015; Mano *et al.*, 2016). The VNP is then restricted to CP policies, while generating embedding solutions which must best satisfy the SP's requirements. On the other hand, the few early heuristics-based solutions on the multicloud VNE problem have generally assumed that CPs would disclose some private information (Leivadeas *et al.*, 2013; Samuel *et al.*, 2013). Some others aim only at minimizing the resource provisioning price for the SP during the splitting phase (Dietrich *et al.*, 2015; Mano *et al.*, 2016), regardless of the performance and QoS of resulting embedded VNR segments.

In this paper, we propose a QoS-based splitting strategy for a VNRs embedding across multiple IaaS providers. Our approach considers resource and QoS constraints based on SP's requirements, with the purpose of splitting efficiently the VNRs and improving the performance of resulting VNR segments that are mapped onto the selected intracloud infrastructures. The key contributions of this work are as follows:

— We have designed a multicloud VNRs splitting approach based on the limited information disclosed by the CPs and the dynamic and heterogeneous nature of the substrate

network. In addition, for a better efficiency of our splitting strategy, we have enriched the visibility of the VNP on the multicloud environment, by taking advantage of certain information related to network topologies that is not treated as confidential, such as CPs' Point-of-Presence (PoP)(Dietrich *et al.*, 2015);

— We use an Integer Linear Programming (ILP) model to formalize the proposed strategy as a maximization problem with constraints. Resource provisioning costs are defined based on the availability of supplied resources and the performance guarantees advertised by the CPs;

— Taking into account the NP-hard nature of the VNRs splitting phase problem, we propose a Tabu Search (TS) algorithm (Glover, 1989, 1990), in order to solve large instances of the problem in polynomial computing time;

— To handle the VNR segments mapping phase, we have adopted a multi-objective intracloud mapping approach for each selected CP, in order to minimize mainly the overall delay. The adopted approach follows the work of (Larumbe et Sanso, 2013), which formalizes the mapping problem as a Mixed-Integer Linear Programming (MILP).

The rest of the paper is organized as follows: Section 5.2 discusses relevant related work. Section 5.3 describes the multicloud VNE problem and the proposed embedding framework. Section 5.4 presents the mathematical formulation related to the VNRs splitting problem. Section 5.5 presents the adaptation of the proposed algorithm. Performance evaluation and simulation results are presented in Section 5.6. Finally, conclusion and future works are highlighted in Section 5.7.

5.2 Related work

VNE represents the main resource allocation challenge in cloud infrastructures virtualization (Belbekkouche *et al.*, 2012; Manvi et Shyam, 2014). It is known to be an NP-hard problem (Chowdhury *et al.*, 2012; Zhang *et al.*, 2016a). The problem consists on a simultaneous and optimized mapping with constraints of virtual nodes resources and virtual links (VLs) resources onto the substrate networks, which is generally reduced to the NP-hard multiway separator problem (Fischer *et al.*, 2013). As a result, various heuristic-based algorithms have been proposed, in order to satisfy economic benefits, resource utilization-efficiency, energy-efficiency, survivability and QoS aspects. Indeed, exact approaches are not scalable and can generally only be applicable in small sized scenarios (Dietrich *et al.*, 2015; Houidi *et al.*, 2011, 2015; Mechtri *et al.*, 2017).

Here we discuss related work on VNE over a single-cloud network and VNE over a multicloud network.

5.2.1 VNE over a single-cloud network

Many algorithms, mostly based on (meta) heuristic approaches, have been proposed for the single-domain VNE optimization problem. Some works tend to solve the problem by providing a certain coordination between the node mapping stage and the link mapping stage (Chowdhury *et al.*, 2012). (Hesselbach *et al.*, 2016) recently proposed a new path algebra-based embedding strategy that coordinates in a single step the mapping of nodes and links.

Dynamic methods, which support remappings of resources during the life-time of requests, have also been suggested. (Ayoubi *et al.*, 2016) proposed a Tabu-based framework for reliable VNE with migration, which consists in availability-aware resource allocation and reconfiguration. (Rahman et Boutaba, 2013) proposed a fast re-embeddings strategy based on a hybrid policy heuristic that aims at network survivability. (Zhang *et al.*, 2014) took a step further by introducing a first-fit-based approach to support opportunistic resource sharing among virtual nodes and VLs. Because most of these approaches may suffer from improper load balancing and link under-utilization, (Khan *et al.*, 2016) proposed a proactive and reactive multi-path link embedding approach to achieve the VNE survivability, while minimizing both resource redundancy and path splitting overhead.

Other works introduce multi-objective approaches to resolve the problem. (Larumbe et Sansò, 2012; Larumbe et Sanso, 2013) proposed an efficient multi-objective model for mapping virtual resources into various locations of cloud data centers. Their solution is based on a TS approach that allows CPs to minimize delay, environmental and resource operating costs. In the same way, the authors (Melo *et al.*, 2013) proposed an ILP model to solve the online VNE problem in order to minimize the resource consumption, while performing load balancing. Their work was extended in (Melo *et al.*, 2015), by considering in addition the minimization of energy consumption. (Houidi *et al.*, 2015) also proposed a multi-objective fault-tolerant VNE algorithm formalized as a MILP, which takes into account constraints related to power consumption, resource availability and load balancing.

These intra-domain VNE solutions, although they may be optimal, can not be properly applicable to a multi-domain VNE problem. They are performed by the CP based on a complete knowledge of its substrate network topologies and available resource distributions. As a result, upon splitting the VNRs, they can only be used by each selected CP for the VNR segments intracloud mapping phase.

5.2.2 VNE over a multicloud network

VNE over a multi-domain has not been intensively studied. Research works have essentially focused on the VNRs splitting phase, since existing solutions related to the single-domain VNE problem are usually adopted for the mapping phase.

(Houidi *et al.*, 2011) were one of the first authors working in this research field. They combined an exact ILP method with a heuristic algorithm based on recursive max-flow min-cut to find the optimal VNRs splitting across multiple CPs. (Samuel *et al.*, 2013) introduced a distributed protocol for VNE problem in multiple substrate networks. Their approach aims at coordinating the participating CPs through competitive pricing mechanisms, in order to maximize their revenue. The proposal of (Leivadeas *et al.*, 2013) is one of the few performance and efficiency-based approaches for VNRs splitting. The authors proposed a hierarchical framework where the VNRs splitting problem is solved through an Iterated Local Search (ILS) algorithm. However, even though costs are not randomly generated as with (Houidi *et al.*, 2011), their approach does not respect certain information concealed by the CPs (Dietrich *et al.*, 2015). Also, their resource provisioning costs definition lacks on the heterogeneous nature of the substrate network. (Mechtri *et al.*, 2017) proposed a heuristic algorithm based on graph decomposition into topology patterns and bipartite graph matching, in order to solve the resources mapping problem in distributed and hybrid cloud environments.

On the other hand, some works have considered the limited level of access to information of the VNP before proceeding to the splitting phase. (Dietrich *et al.*, 2015) have studied the suboptimality of multi-domain VNE with limited information disclosure, whose the efficiency is compared with the "best-case" scenario where the complete network topology and resource availability information is accessible by the VNP. Authors (Mano *et al.*, 2016) proposed also a novel optimization method restricted to CPs' private information, which uses a secure multi-party computation (MPC) to generate minimal operations that minimize inter-domain VNE prices. However, both proposals are only focus on VNRs splitting at the lowest price for the SP, without considering performance and QoS of embedded VNR segments.

Based on the literature, most VNRs splitting strategies across multiple CPs are only based on exact methods, and thus can only scale up to topologies with small sizes (Dietrich *et al.*, 2015). Some others based on heuristics and using a more sophisticated splitting scheme (Leivadeas *et al.*, 2013; Samuel *et al.*, 2013) assume that CPs would share some private information with the VNP. Not to mention that most proposals, even though they are restricted to the domain-privacy of CPs (Dietrich *et al.*, 2015; Gong *et al.*, 2016; Li *et al.*, 2016; Mano *et al.*, 2016), tend only to minimize the resource provisioning price for the SP. This may not give opportunities to the SP to select CPs based only on desired performance and QoS.

5.3 Multicloud VNE problem

In this section, we first briefly describe the problem of VNE across multiple CPs. Then, we present an overview of the proposed framework, where a hierarchical approach is generally adopted to decompose the global problem into the VNRs splitting phase and the VNR segments mapping phase.

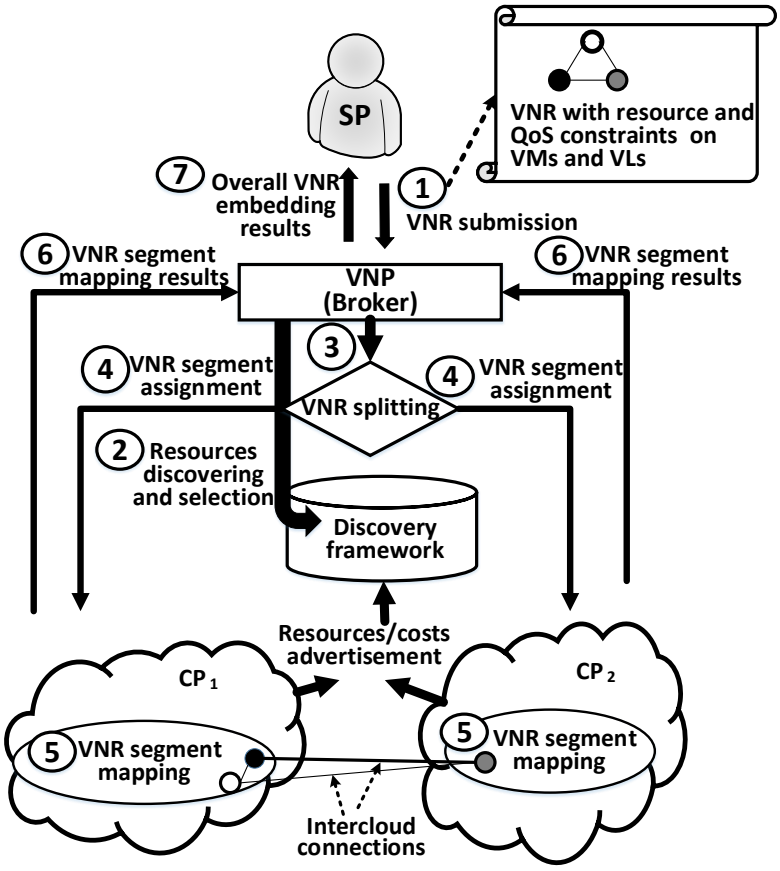

Figure 5.1 Multicloud VNE process

5.3.1 Problem description

The global multicloud VNE process is described in Figure 5.1. The problem consists in efficiently embedding VNRs onto multiple cloud networks owned by independent CPs. VNRs are submitted in a high level of abstraction by the SP, as a set of virtual nodes interconnected by VLs representing the exchanged traffic between virtual nodes (Fischer *et al.*, 2013). Virtual nodes and VLs are specified with constraints related to required resources and QoS criteria

(e.g., processing, memory, storage, bandwidth, QoS parameters, geographic footprint, etc.), that the embedding process has to satisfy. The SP will generally rely on the VNP, which acts as a broker to discover and select a set of advertised resources, assembled from multiple CPs and stored in a discovery framework (Lv *et al.*, 2010). Heterogenous resources are then allocated to host virtual nodes (e.g. VMs) in specific substrate nodes (e.g. data centers), and to route VLs onto substrate paths. In most cases, to meet the SP's requirements, the VNP will need to optimally split the VNRs among eligible CPs. The resulting VNR segments are then mapped by each selected CP, and are subsequently interconnected via appropriate intercloud links (Leivadeas *et al.*, 2013).

5.3.2 Multicloud VNE framework

Here, an analysis of the resource discovery framework is presented, followed by a summary description of the approaches proposed to solve each of the splitting and mapping phases.

5.3.2.1 Resource discovery framework and assumptions

The VNP visibility on the multicloud substrate network is essentially decisive at the splitting phase. An effective splitting strategy should conduct a thorough investigation into the discovery of resource information, in order to avoid inefficient embedding solutions.

A CP typically classifies its substrate nodes into different categories (e.g. (Amazon, 2017)), each of them having common set of functional attributes and associated with a set of computational resource types (Leivadeas *et al.*, 2013). Node functional attributes define characteristics and properties related to the node type, operating system, virtualization environment, geographic footprint, QoS parameters, etc. Computational resource types can be the CPU, disk space, memory, etc., each of them associated with a certain capacity that the CP keeps dynamically updated (Houidi *et al.*, 2011). Similarly, substrate links are also classified into different types (e.g. VLAN, L3/L2 VNP, etc.) and are associated with a bandwidth capacity dynamically updated too. However, as previously mentioned, detailed information such as substrate network topologies, router-level connectivity, transit network topologies, number of data centers located in a cloud and their interconnectivity, number of instances of available resources and their utilization, are not disclosed to the VNP. Nevertheless, some information about transit networks (PoPs) of CPs can be accessible in a high level of abstraction (Dietrich *et al.*, 2015; Doverspike *et al.*, 2010), allowing the VNP to expand its limited view on the substrate multicloud network. Those information can include an oversimplified view of PoP networks, traffic statistics and some performance guarantees for a link such as delay bounds (Larumbe et Sanso, 2013; Lv *et al.*, 2010). A CP can also supply the maximum or

the minimum capacity of a resource it can offer or guarantee (Chaisiri *et al.*, 2012), without revealing details about the distribution and utilization of these resources inside its intracloud network.

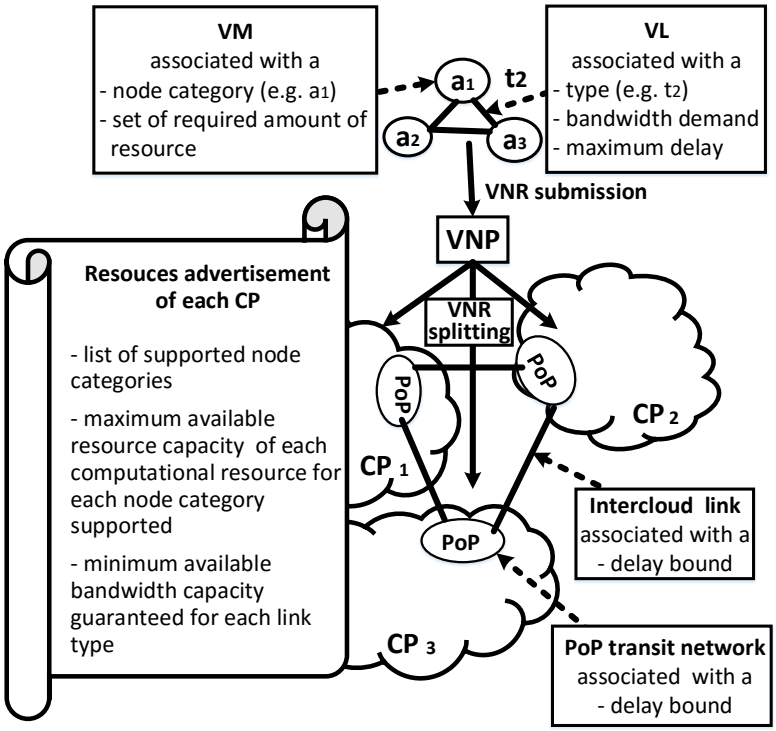

Figure 5.2 VNRs splitting phase based on the resource discovery framework

Based on these observations, in our framework the following information are considered accessible by the VNP: (i) the maximum available capacity of each computational resource type a CP can offer for each category of node; (ii) the minimum available bandwidth capacity a CP can guarantee in its intra-domain for each type of link; (iii) the estimated delay bound on an intercloud link and for transiting through a PoP network.

Without loss of generality, we consider that virtual nodes are virtual machines (VMs) packaged by the SP. A PoP network is also represented in a high level of abstraction as a macro node associated with a transiting delay bound. For sake of simplicity, we only consider one PoP network per CP. CPs are interconnected in a half mesh connectivity and we did not consider in this work the path splitting scenario, as the VNP does not know the residual bandwidth capacity of intercloud paths. Note that CPs may also advertise the unit monetary cost of resources they offer in the discovery framework. But, in this present work, the expenditure minimization for the SP is not our interest. We only focus on an efficient VNE that targets the performance and the QoS of split VNRs.

5.3.2.2 Multicloud VNRs splitting phase

The multicloud VNRs splitting phase is illustrated in Figure 5.2. Upon receiving an incoming VNR, the VNP relies on the above disclosed information to identify eligible substrate resources able to fulfill the SP's requirements. To simplify this matching step, we assume that VMs and VLs are classified and specified with characteristics at the same level as substrate nodes and substrate links respectively. Furthermore, we consider that VLs are associated with a maximum communicating delay, in order to allow the QoS fulfilment for delay-sensitive applications. Resources required by a VNR may be supplied by more than one CP, at different provisioning costs. In our framework, the VNP will perform a VNRs splitting strategy, by using a TS-based approach to evaluate the most cost-effective resource provisioning depending on their availability, while considering QoS restrictions specified in the request. Thereafter, the VNP sends the resulting VNR segments to the selected CPs so they can proceed to the intracloud mapping phase.

TS metaheuristic is adopted in this framework mainly due its efficiency in achieving optimal or near-optimal solutions for various optimization problems that deal with scalability issues.

5.3.2.3 Intracloud VNR segments mapping phase

Each CP receiving a particular VNR segment maps it onto its infrastructure by using an appropriate intracloud mapping method. In our framework, we adopt a multi-criteria solution based on the principles stated by (Larumbe et Sansò, 2012; Larumbe et Sanso, 2013). In this approach, a selected CP must map in the best case all the assigned request segments, by using a multi-objective MILP model. The latter aims to minimize the total embedding cost (including resource, traffic and environmental costs), while providing the best possible QoS by reducing mainly the overall traffic routing delay. However, we have re-stated the approach in order to remain consistent with the level of granularity we have considered in this work, which is limited to data centers and not to physical servers hosted into the data centers.

5.4 System modeling and mathematical formulation

In this section, the notation used to describe the multicloud VNRs splitting problem is presented, followed by the mathematical formulation related to the proposed strategy. The modeling, notation and formulation related to the intracloud VNR segments mapping problem are available in Appendix A.

5.4.1 Notation

The notation is composed of sets, parameters and variables describing the multicloud substrate network, the VNR, the costs definition and the decision variables:

Global sets: Let A designate the set of node categories, R the set of computational resource types and T the set of link types.

Multicloud substrate network: The substrate network is modeled as a weighted undirected graph $G^S = (N^S, L^S)$, where N^S is the set of substrate nodes and L^S the set of substrate links (intercloud links). Let I designate the set of CPs and Θ the set of all PoP nodes (transit networks). Note that N^S includes I and Θ, and PoP nodes are only used to access the network of a CP or to transit through it. The maximum available capacity of resource $r \in R$ that CP i can offer for nodes of category $a \in A$ is denoted by Q_{rai} ($Q_{rai} \in \mathbb{N}$). The minimum available bandwidth capacity that CP i can guarantee in its intracloud network for links of type $t \in T$ is denoted by B_{ti} ($B_{ti} \in \mathbb{N}$). The average delay bound of intercloud link $e \in L^S$ and for transiting through PoP network $p \in \Theta$ are respectively denoted by d_e and d_p. Let $\mathcal{P}_{ij}$ designate the set of all paths between CP i and CP j. We denoted by $\mathcal{O}_\varphi$ the set of all PoP transit networks in Θ spanned by path $\varphi \in \mathcal{P}_{ij}$.

VNR modeling: The VNR is also modeled as a weighted undirected graph $G^V = (N^V, L^V)$, where N^V is the set of VMs and L^V the set of VLs representing the inter-VMs traffic. Let N_a^V denote the set of VMs defined in category of node $a \in A$ ($\bigcup_{a \in A} N_a^V = N^V$). The required amount of resource $r \in R$ by VM v is denoted by q_{rv} ($q_{rv} \in \mathbb{N}_\mathbb{1}$). Let q_r^{min} and q_r^{max} designate respectively the lowest and the highest amount of resource $r \in R$ ever requested in a VNR. Let L_t^V denote the set of VLs of type $t \in T$ ($\bigcup_{t \in T} L_t^V = L^V$). The bandwidth demand and the maximum allowed traffic delay of VL l are respectively denoted by $b_l (b_l \in \mathbb{N}_\mathbb{1})$ and δ_l. This maximum delay adds constraints on the assignment of communicating VMs, in order to avoid delay violations on VLs.

Cost parameters: We denote by $\mathcal{C}_{rai}^N$ the node provisioning cost of CP i for nodes of category $a \in A$ for resource of type $r \in R$. Let $\mathcal{C}_{ti}^L$ designate the intracloud link provisioning cost of CP i for links of type $t \in T$, and $\mathcal{C}_{t\varphi}^L$ the intercloud link provisioning cost of path $\varphi \in \mathcal{P}_{ij}$, $i, j \in I$, for links of type $t \in T$.

Other parameters: We defined other parameters whose equations are stated in section 5.4.2.1. Let Q_{ra}^{max} denote the highest available capacity of resource $r \in R$ supplied among all CPs for nodes of category $a \in A$. Let B_t^{max} denote the highest available bandwidth capacity guaranteed among all CPs for links of type $t \in T$. Let w_{rv} designate the weight of resource

$r \in R$ required by VM v. Let α and β denote respectively the node demand weight and the link demand weight in the total VNRs splitting cost.

Decision variables: Let X_{vi} denote the binary variable set to 1 if VM v is assigned to CP i and 0 otherwise. Let $Y_{l\varphi}$ denote the binary variable set to 1 if VL l is assigned to path $\varphi \in \mathcal{P}_{ij},\ i,j \in I,\ t \in T$, and 0 otherwise.

5.4.2 VNRs splitting problem formulation

Our splitting strategy aims to design a VNE method with the best performance of resulting VNR segments, while satisfying resources and QoS requirements. To this end, we define the intracloud resource provisioning cost based on the availability of supplied resources. The provisioning cost of an intercloud path is stated as a function of the intracloud link provisioning cost of the two corresponding endpoints CPs, and the number of PoP transit networks the path spans.

5.4.2.1 Multicloud resource provisioning costs definition

We first define by equation (5.1) the parameter Q_{ra}^{max}. The node provisioning cost per CP is then given by equation (5.2), as the maximum available capacity of each resource the CP can offer for each category of nodes, normalized into the range [0 1] by dividing it by Q_{ra}^{max} (or by the value 1 to avoid division by zero):

$$Q_{ra}^{max} = \max_{i \in I} \left(Q_{rai}\right), \qquad \forall\ r \in R,\ a \in A \tag{5.1}$$

$$C_{rai}^{N} = \frac{Q_{rai}}{\max\left(Q_{ra}^{max}, 1\right)}, \qquad \forall\ r \in R,\ a \in A,\ i \in I \tag{5.2}$$

The parameter w_{rv} is given by the following, where the values of q_r^{max} and q_r^{max} are estimated and kept updated by the VNP based on previous statistics:

$$w_{rv} = \frac{q_{rv} - q_r^{min}}{q_r^{max} - q_r^{min}}, \qquad \forall\ r \in R,\ v \in N^V, \quad q_r^{min}, q_r^{max} \in \mathbb{N}_{\mathbb{1}},\ q_r^{min} \leq q_{rv} \leq q_r^{max} \tag{5.3}$$

The parameter B_t^{max} is defined by equation (5.4). The intracloud link provisioning cost per CP is then given by equation (5.5), as the minimum available bandwidth capacity the CP can guarantee for each link type, normalized into the range [0 1] by dividing it by B_t^{max} (or by the value 1 to avoid division by zero):

$$B_t^{max} = \max_{i \in I} \left(B_{ti}\right), \qquad \forall\ t \in T \tag{5.4}$$

$$C_{ti}^{L} = \frac{B_{ti}}{\max(B_t^{max}, 1)}, \qquad \forall\ t \in T,\ i \in I \tag{5.5}$$

The intercloud link provisioning cost of each path between a pair of CPs is given by equation (5.6), as the minimum intracloud link provisioning cost between the two endpoint CPs of the path, divided by the number of PoP transit networks spanned:

$$C_{t\varphi}^{L} = \frac{\min\left(C_{ti}^{L}, C_{tj}^{L}\right)}{\max\left(\left|\mathcal{O}_{\varphi}\right|, 1\right)}, \qquad \begin{array}{l} \forall\ t \in T,\ i, j \in I, \varphi \in \mathcal{P}_{ij}. \\ \text{with } \forall \varphi \in \mathcal{P}_{ii},\ i \in I,\ \mathcal{O}_{\varphi} = \emptyset \end{array} \tag{5.6}$$

Note that if the two endpoints of a path represent the same CP (i.e. i is equal to j), equation (5.6) will be equivalent to the corresponding CP's intracloud link provisioning cost stated in equation (5.5). Since our objective function is a cost maximization, equation (5.6) will guide a solution to prefer the intracloud paths, otherwise the intercloud paths spanning the least possible PoP transit networks.

Since the number of VLs is usually much higher than the number of interconnected VMs, we need to balance the node cost and the link cost in our objective function. To this end, we define the parameters α and β as follows:

$$\alpha = \frac{\left|L^V\right|}{|N^V| + |L^V|}, \qquad \beta = 1 - \alpha \tag{5.7}$$

5.4.2.2 ILP formulation

The objective function of the proposed ILP model is expressed as follows:

MAX

$$\alpha \sum_{i \in I} \sum_{r \in R} \sum_{a \in A} \sum_{v \in N_a^V} w_{rv} C_{rai}^{N} X_{vi} \quad + \beta \sum_{i \in I} \sum_{j \in I} \sum_{\varphi \in \mathcal{P}_{ij}} \sum_{t \in T} \sum_{l \in L_t^V} C_{t\varphi}^{L} Y_{l\varphi,} \tag{5.8}$$

The first term of the objective function represents the total node provisioning cost for assigning VMs to CPs. The second term represents the total link provisioning cost for assigning VLs to intracloud paths (i.e where i is equal to j) or intercloud paths (i.e. where i is different from j). Note that the objective function is maximized since the evaluated cost represents a benefit for the SP.

The model is subjected to the following constraints:

$$\sum_{i \in I} X_{vi} = 1, \qquad \forall\ v \in N_a^V,\ a \in A \tag{5.9}$$

Constraint (5.9) ensures that each VM must be assigned to exactly one CP.

$$Y_{l\varphi} \leq \frac{X_{ui} + X_{vj}}{2}, \qquad \begin{array}{l} \forall\ l = uv \in L_t^V,\ t \in T, \qquad u, v \in N_a^V (v \neq u),\ a \in A, \\ \varphi \in \mathcal{P}_{ij},\ i, j \in I \end{array} \tag{5.10}$$

Constraint (5.10) states the binary value of the variable $Y_{l\varphi}$ for each VL between two communicating VMs.

$$\sum_{i \in I} \sum_{j \in I} \sum_{\varphi \in \mathcal{P}_{ij}} Y_{l\varphi} = 1, \qquad \forall\ l \in L_t^V,\ t \in T \tag{5.11}$$

Constraint (5.11) ensures that each VL must be assigned to exactly one intracloud path, otherwise to an unique intercloud path.

$$\sum_{v \in N_a^V} q_{rv} X_{vi} \leq Q_{rai}, \qquad \forall\ r \in R,\ a \in A,\ i \in I \tag{5.12}$$

Constraint (5.12) ensures that the total amount of a resource required by all VMs assigned to a CP, must not exceed the maximum available capacity of the resource offered.

$$\sum_{i \in I} \sum_{j \in I} \sum_{\varphi \in \mathcal{P}_{ij}} d_\varphi Y_{l\varphi} \leq \delta_l, \forall\ l \in L_t^V,\ t \in T \tag{5.13}$$

where

$$d_\varphi = \sum_{e \in \varphi} d_e\ + \sum_{p \in O_\varphi} d_p, \qquad \forall\ \varphi \in \mathcal{P}_{ij},\ i, j \in I \tag{5.14}$$

Constraint (5.13) ensures that the total delay on a path to which a VL is assigned must be less than the maximum delay allowed for the VL.

$$X_{vi} \in \{0, 1\} \qquad \forall\ v \in N_a^V,\ a \in A,\ i \in I \tag{5.15}$$

$$Y_{l\varphi} \in \{0, 1\} \qquad \forall\ l \in L_t^V,\ t \in T, \qquad \varphi \in \mathcal{P}_{ij},\ i, j \in I \tag{5.16}$$

Constraints (5.15) and (5.16) express the binary domain of each variable.

5.5 The proposed TS splitting algorithm

In this section, we present the basic principles of the TS algorithm, followed by the proposed adaptation of the metaheuristic, named *TS_Split*, in order to efficiently solve the VNRs splitting problem for large sized instances.

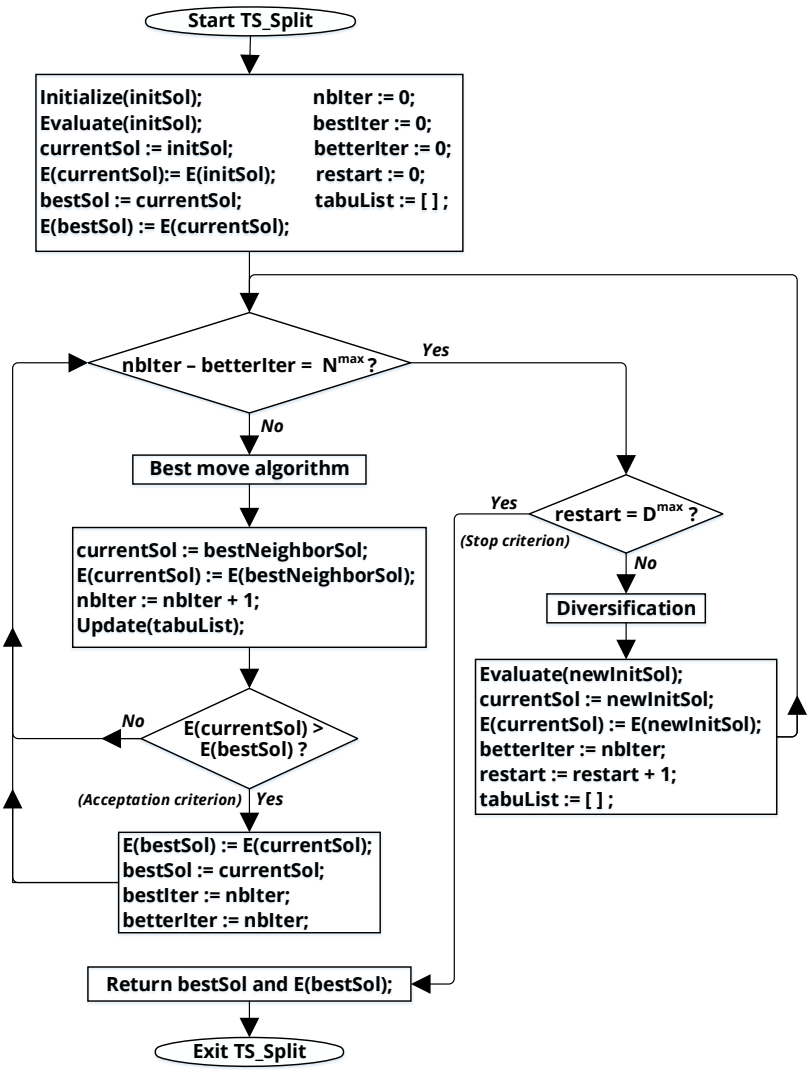

Figure 5.3 The proposed TS algorithm

5.5.1 Basic principles of TS

The TS algorithm uses a local search (LS) method which explores the solution space by moving iteratively from a solution S to the best solution in the neighborhood of S. Contrary to classical descent methods, to overcome local optima and cycling problems, the TS method

allows moves that deteriorate the actual best solution and stores temporarily forbidden moves in a tabu list. However, a tabu move can occasionally be allowed if it satisfies a specific aspiration criterion. The search stops whenever a given stop criterion is satisfied. The TS method can also be enhanced by using the adaptive memory mechanisms.

5.5.2 TS_Split algorithm

The proposed algorithm is presented in Figure 5.3. It combines a LS process and a long-term memory mechanism. *TS_Split* starts with an initial solution *initSol*. Then, at each iteration, the LS process tries to improve the best solution *bestSol*, by only accepting current improving solutions *currentSol*. If *bestSol* has not been improved after N^{max} iterations, the long-term memory mechanism is then performed. The latter generates a new initial solution *newInitSol*, from which the LS process restarts the search in order to find a better solution. *TS_Split* algorithm stops after D^{max} runs of the *Diversification* mechanism. The parameters of the algorithm are defined in Section 5.6.2 as results of parameterization tests. The solution space, the initial solution, the LS process, as well as the long-term memory mechanism are detailed in the following.

5.5.2.1 Solution space

A solution S of *TS_Split* is an assignment of each VM of a VNR to a unique CP, and the assignment of each VL to a unique intracloud or intercloud path. Thus, a solution is determined by the setting of variables X_{vi} and $Y_{l\varphi}$ that satisfies the constraints of the problem, particularly the ones stated in (5.12) and (5.13). To avoid unfeasible solutions, two negative penalty costs $P^N(S)$ and $P^L(S)$, related respectively to the possible violation of constraints (5.12) and (5.13), are added to the objective function. Thereby, the cost of a solution S is defined by the evaluation cost $E(S)$, which includes the intrinsic cost $C(S)$ related to the objective function in (5.8), plus the penalty costs $P^N(S)$ and $P^L(S)$. At each iteration, the LS process accepts better solutions S that improves $E(S)$.

5.5.2.2 Initial solution

In order to generate the initial solution *initSol*, three methods were tested: (i) the highest node provisioning cost method, where VMs are sorted in descending order of their average weight of all requested resources and then assigned to the first suitable CP that has the highest node provisioning cost; (ii) the single-CP assignment method, where all VMs are assigned to a randomly selected CP; (iii) the random generating of initial solution, where

each VM is randomly assigned to a CP and each VL is assigned to the shortest path between the corresponding pair of CPs. Note that any of these methods can guaranty a feasible *initSol* simultaneously for VMs and VLs. Best solutions were noticed with the third method. Therefore, a totally random initial solution were considered in our algorithm.

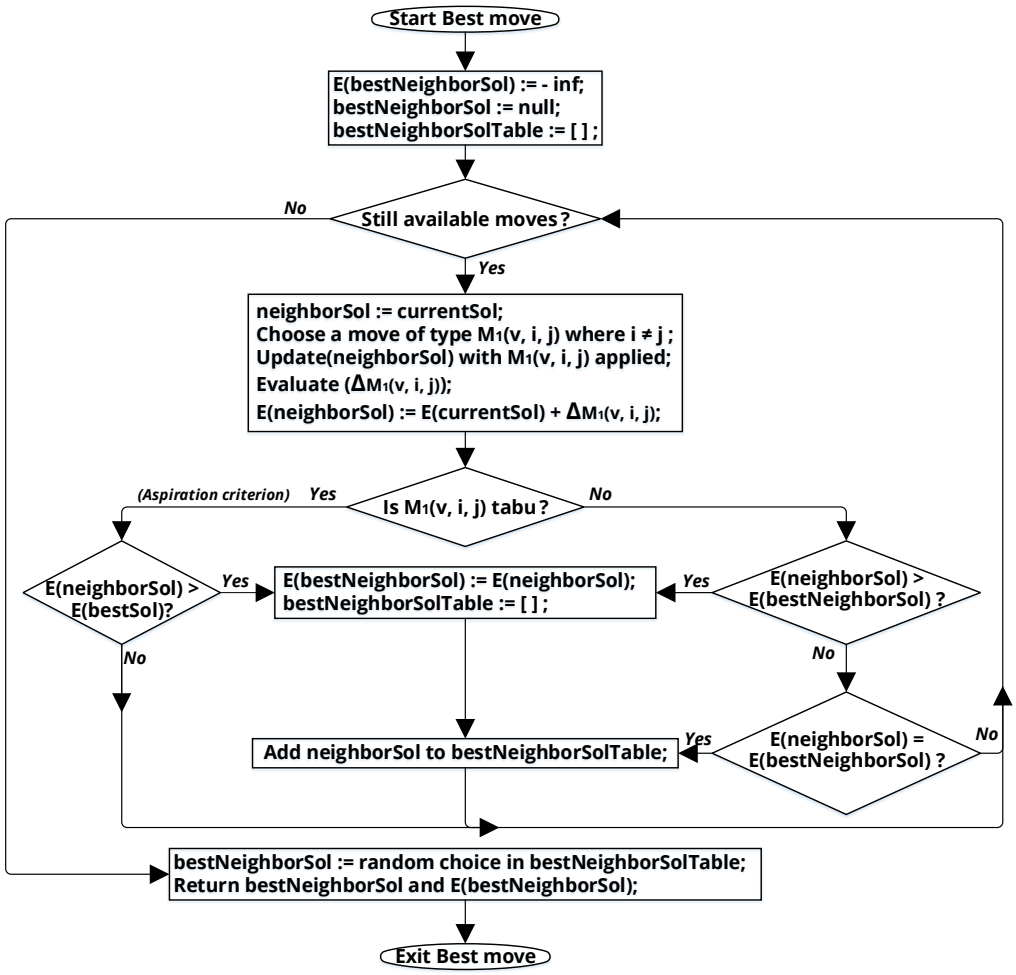

Figure 5.4 The best move algorithm

5.5.2.3 LS process

Each iteration of the LS process consists in moving from a current solution *currentSol* to its best neighbor solution, by choosing the best move to apply to the current solution. In the following, we describe our best move algorithm, which defines the best neighbor *bestNeighborSol* of *currentSol*, and how the list of temporarily tabu (forbidden) moves *TabuList* is updated after selecting the best move.

a) Best move algorithm Two different types of moves were examined in this work:

- $M_1(v, i, j)$ which moves VM v from its actual assigned CP i to a different CP j;

- $M_2(v_1, v_2)$, which swaps VM v_1 and VM v_2, respectively assigned to different CPs;

The swapping move $M_2(v_1, v_2)$ was computationally more expensive than the simple move $M_1(v, i, j)$. With the established parameters of our algorithm, the simple move was globally as efficient as the swapping move in finding near-optimal solutions with a much shorter calculation time. For this reason, only the simple move was applied in our algorithm.

The best move algorithm is presented in Figure 5.4. It explores the entire neighborhood of *currentSol* while there is still available moves. The move $M_1(v, i, j)$ that generates the highest evaluation cost $E(bestNeighborSol)$, depending on its tabu status and the aspiration criterion, is chosen. The aspiration criterion allows a tabu move if only the latter improves *bestSol*. Also, instead of selecting the first (or the last) best move after evaluating all the available moves, our algorithm considers all the moves that generate the same highest $E(bestNeighborSol)$ and randomly selects one of them.

Furthermore, instead of evaluating the entire objective function and the penalty costs for each neighbor solution, only the evaluation cost difference $\Delta_{M_1(v,i,j)}$ generated by the move from the current solution to its neighbor solution is calculated. This approach is more complicated to implement but it decreases significantly the computing time, by reducing the complexity order of cost calculation from $O(n^2)$ to $O(n)$, with n representing the number of VMs in a request.

Let *neighborSol* denote a neighbor solution of *currentSol* after applying a move $M_1(v, i, j)$ on *currentSol*. The gain $\Delta_{M_1(v,i,j)}$, with $v \in N_a^V,\ a \in A,\ i, j \in I,\ j \neq i$, is defined as follows:

$$\Delta_{M_1(v,i,j)} = \alpha \left(\Delta^N_{M_1(v,i,j)} + \Delta^{P^N}_{M_1(v,i,j)} \right) + \beta \left(\Delta^L_{M_1(v,i,j)} + \Delta^{P^L}_{M_1(v,i,j)} \right) \tag{5.17}$$

$\Delta^N_{M_1(v,i,j)}$ represents the intrinsic node cost difference from *currentSol* to *neighborSol*, given by:

$$\Delta^N_{M_1(v,i,j)} = \sum_{r \in R} w_{rv} \left(C^N_{raj} - C^N_{rai} \right) \tag{5.18}$$

$\Delta^{P^N}_{M_1(v,i,j)}$ represents the node penalty cost difference from *currentSol* to *neighborSol*, given by:

$$\Delta^{P^N}_{M_1(v,i,j)} = -10^5 \times \sum_{r \in R} \left(\begin{array}{l} \max \left(0, \sum\limits_{\substack{w \in N^V_a \\ w \neq v}} (q_{rw} X_{wj}) + q_{rv} - Q_{raj} \right) \\ + \max \left(0, \sum\limits_{\substack{w \in N^V_a \\ w \neq v}} (q_{rw} X_{wi}) - q_{rv} - Q_{rai} \right) \end{array} \right) \tag{5.19}$$

$\Delta^{L}_{M_1(v,i,j)}$ represents the intrinsic link cost difference from *currentSol* to *neighborSol*, given by:

$$\Delta^{L}_{M_1(v,i,j)} = \sum_{\substack{l = vw \in L^V_t \\ w \in N^V \\ w \neq v}} \left(C^L_{t\varphi_1} - C^L_{t\varphi_2} \right) + \sum_{\substack{l = wv \in L^V_t \\ w \in N^V \\ w \neq v}} \left(C^L_{t\varphi_3} - C^L_{t\varphi_4} \right), \tag{5.20}$$

where φ_1, φ_2, φ_3 and φ_3 are respectively defined as follows:

$$\begin{aligned} \varphi_1 &= \left\{ \varphi \in P_{jk_w} \mid Y^+_{l\varphi} = 1, \forall l = vw \in L^V_t, w \in N^V, w \neq v \right\} \\ \varphi_2 &= \left\{ \varphi \in P_{ik_w} \mid Y^-_{l\varphi} = 1, \forall l = vw \in L^V_t, w \in N^V, w \neq v \right\} \\ \varphi_3 &= \left\{ \varphi \in P_{k_w j} \mid Y^+_{l\varphi} = 1, \forall l = wv \in L^V_t, w \in N^V, w \neq v \right\} \\ \varphi_4 &= \left\{ \varphi \in P_{k_w i} \mid Y^-_{l\varphi} = 1, \forall l = wv \in L^V_t, w \in N^V, w \neq v \right\} \end{aligned}$$

$Y^+_{l\varphi}$ and $Y^-_{l\varphi}$ represent respectively the new and the old assignment of VL l on the given path φ. k_w represents the CP to which VM w is assigned, i.e.:

$$k_w = \left\{ i \in I \mid X_{wi} = 1, \forall\ w \in N^V \right\}$$

$\Delta^{P^L}_{M_1(v,i,j)}$ represents the link penalty cost difference from *currentSol* to *neighborSol*, given by:

$$\Delta^{P^L}_{M_1(v,i,j)} = -10^5 \times \left(\left| H^L_{neighborSol} \right| - \left| H^L_{currentSol} \right| \right), \tag{5.21}$$

where $H^L_{currentSol}$ and $H^L_{neighborSol}$ represent the set of VLs whose the resulting assignment, respectively in the current solution and the neighbor solution, does not respect the constraint

(5.13). Note that at each movement, each VL is computed to be assigned to the shortest path (in terms of PoP networks spanned) between a chosen pair of CPs that does not violate its maximum delay.

b) Tabu list To avoid repeated solutions that keep returning to a local optimum, once the best move $M_1(v, i, j)$ is applied, the tabu list is updated by adding the pair (v, i) to $TabuList$, which forbids the heuristic to assign VM v to CP i for S^{TL} iterations. Details related to the formula we use to define the length S^{TL} of the tabu list are given in Section 5.6.2.

5.5.2.4 Long-term memory mechanism

The *Diversification* mechanism aims at generating new solutions from so far unexplored regions. To this end, after each iteration of the LS process, the resulting solution is kept in a statistics memory table, which stores the number of times each VM has been assigned to a CP. Then, based on the least explored assignment criterion, the *Diversification* mechanism creates a new initial solution $newInitSol$ from which a new exploration restarts. An *intensification* method around the neighborhood of the best solution has also been tested in this work, but the *Diversification* mechanism has given us far better solutions for such an optimization problem.

5.6 Numerical Results

In this section, the efficiency of the proposed TS-based VNRs splitting strategy is evaluated through simulations. To this end, two main series of experiments with different scenarios considered in each of them are conducted: *Experiment 1*, where the performance of *TS_Split* algorithm is evaluated in terms of quality of solutions obtained and computing time; and *Experiment 2*, where the performance of the proposed VNRs splitting strategy is evaluated according to several performance metrics, such as acceptance rate, execution time, splitting rate and delay.

The proposed approach is implemented in C++ under Visual Studio 14. All simulations are run on a single server with an Intel Core i7 CPU at 4 GHz and 32 GB of RAM.

5.6.1 Experiments setup

Each of *Experiment 1* and *Experiment 2* is conducted considering both the strategy with 5 CPs and 10 CPs. We have also stated two scenarios in each experiment, respectively for small sized instances and large sized instances.

— ***Experiment 1***: Here we first consider a *Scenario 1* which compares the proposed *TS_Split* algorithm with the exact method. The latter, named *Exact_Split*, is used to calculate the optimal solution of the proposed ILP and is implemented in AMPL 64 bits with the CPLEX 12.6.3 solver. In *Scenario 1*, we consider 14 instances of small sized VNRs, in which the number of VMs increases with an incremental 5-step, respectively from 5 to 70. *TS_Split* algorithm is executed 10 times for each instance. Then, the proposed algorithm is evaluated for large sized VNRs in a *Scenario 2*, by also running 10 times *TS_Split* for 18 instances of VNRs, in which the number of VMs increases with an incremental 25-step, respectively from 75 to 500. In this first experiment, we compare results obtained by also studying different cases where the *Diversification* mechanism and the aspiration criterion are not performed in the proposed algorithm.

— ***Experiment 2***: Here we evaluate the efficiency of our VNRs strategy according to the performance metrics listed above. To this end, we implement in C++ the splitting approach proposed by (Leivadeas *et al.*, 2013), which we name *ILS_Split*. Our algorithm is performed there with the aspiration criterion and the long-term memory. In a *Scenario 3*, we compare *Exact_Split*, *TS_Split* and *ILS_Split* approaches with small sized VNRs (with 5 to 70 VMs), by running 50 simulations with 100 random incoming requests in each simulation. In a *Scenario 4*, we compare only *TS_Split* and *ILS_Split* approaches with large sized VNRs (with 75 to 500 VMs) that the exact method could not solve in polynomial time. We run there 10 simulations with 50 random incoming requests in each simulation. In order to evaluate the acceptance rate metric, for each of *Exact_Split*, *TS_Split* and *ILS_Split* approach, we use the same adopted method of (Larumbe et Sanso, 2013) for the intracloud mapping phase, presented in Appendix A. The adopted approach is implemented for *Scenario 3* with the exact method using the CPLEX solver. For *Scenario 4*, we implement in C++ an iterated descent algorithm as LS method to perform it.

All topologies and features of the substrate network and the VNRs are randomly generated by using a MATLAB program and the parameters set in Table 5.1. Three types of computational resources were considered, which are CPU, memory and disk. VNRs topologies are randomly generated in a partial mesh, with 50% probability of interconnection between VMs. The multicloud substrate network interconnecting all participating CPs is generated based on ISPs topologies (Bhamare *et al.*, 2015; Doverspike *et al.*, 2010). Data centers in each substrate intracloud are interconnected through a backbone network, similar to the NSFNet topology (Amokrane *et al.*, 2013). Substrate nodes are randomly located at different geographic areas and each data center is connected to the backbone network through the closest routers to

its location. The number of substrate nodes in each CP is set to 25, with 20% probability of generating data center nodes and 80% of router nodes. We consider that data centers are heterogeneous (can support different categories of node) with different resource capacities. Residual resource capacities for substrate nodes and substrate links, as well as resources advertised in the discovery framework, are dynamically updated by CPs after a VNR has been mapped or an existing VNR has been released. Details related to the setup of cost parameters used to execute the mapping phase can be consulted in Appendix A.

Table 5.1 Experiments parameters

	Value / distribution interval	
Substrate network		
Number of nodes per CP	25	
Number of links per CP	50 on average	
Degree of nodes interconnectivity per CP	[3, 6]	
Intracloud link delay bound	[0, 3] ms	
Intercloud link / PoP transit delay bound	[0, 25] ms	
	Small sized VNRs	**Large sized VNRs**
DC CPU capacity per node category	[100, 500] cores	[250, 750] cores
DC memory capacity per node category	[1000, 5000] GB	[2500, 7500] GB
DC disk capacity per node category	[10000, 50000] GB	[25000, 75000] GB
Intracloud link bandwidth capacity per link type	[4000, 5000] Mbps	[6000, 8000] Mbps
Virtual network request		
VNR average arrival rate	1 per 100 time units	
VNR lifetime	[1000 100000] time units	
VL maximum delay allowed	[0, 300] ms	
	With 5 CPs	**With 10 CPs**
VM CPU demand	[1, 20] cores	[1, 40] cores
VM memory demand	[1, 200] GB	[1, 400] GB
VM disk demand	[1, 2000] GB	[1, 4000] GB
VL bandwidth demand	[1, 20] Mbps	[1, 20] Mbps

5.6.2 Parameters of *TS_Split*

Preliminary tests performed on small and large instances allowed us to define the optimal parameters of the proposed algorithm, which mostly depend on the problem size. The parameter S^{TL} is defined as $1/3 * \sum_{a \in A} \left|N_a^V\right| * |I_a|$, where I_a represents the set of CPs supporting nodes of category $a \in A$. For small sized VNRs (*Scenario 1* and *Scenario 3*), the parameters N^{max} and D^{max} are defined as follows: $N^{max} = 40 * \sqrt{|N^V| * |I|}$ and $D^{max} = 25 * \sqrt{|N^V| + |I|}$. For large sized VNRs (*Scenario 2* and *Scenario 4*), $N^{max} = 20 * \sqrt{|N^V| * |I| / 2}$ and

$D^{max} = 5 * 10^3 * \sqrt{|I| / |N^V|^2}$. The number of restarts of the *Diversification* mechanism D^{max} is reduced for large instances of the problem due to the fact that the cost improvement is negligible compared to the high increase in the resulting CPU time. Indeed, as VMs are interconnected in partial mesh, with n VMs the number of VLs (in both directions of communication) increases considerably in the range of $\frac{n(n-1)}{2}$. The space of feasible solutions is then significantly reduced because of the higher probabilities of resulting in some violated VLs than in the case of small sized VNRs. Therefore, it becomes useless for the heuristic to explore new areas more than necessary.

5.6.3 Results analysis

The results obtained with each of *Experiment 1* and *Experiment 2* are presented in the following sections.

5.6.3.1 *TS_Split* with Experiment 1

Table 5.2, Table 5.3, Table 5.4 and Table 5.5 show the results obtained with *Scenario 1*, each of them considering the algorithm with the long-term memory and without. For each VNR instance, we report the optimal cost and the CPU time given by the exact method. To demonstrate the efficiency of the proposed heuristic, we present the cost gaps (minimal, maximal and mean) with the optimal solution, expressed in percentage and calculated as $100 * (cost(TS\_Split) - cost(Exact\_Split))/cost(Exact\_Split)$. We also report the minimal, maximal and mean CPU execution time of the algorithm.

The reported results indicate on average zero cost gaps for VNRs with less than 50 VMs with the long-term memory executed. Table 5.2 presents the comparison between *Exact_Split* and *TS_Split* with 5 CPs, where the aspiration criterion is performed. From the table, we can see that *TS_Split* algorithm is able to reach quasi-always the optimal solution when the long-term memory is performed. The minimal cost gap is between 0% and 0.1% and the maximal one is below 1.95%, which shows that the proposed heuristic is very effective in finding optimal or near-optimal solutions. However, cost gaps are higher when the long-term memory is not performed, giving minimal and maximal cost gaps up to 10.31% and 25% respectively. This proves that the *Diversification* mechanism significantly improves the quality of the solutions, allowing the heuristic to effectively explore more promising areas of research.

Table 5.2 Comparison between *TS_Split* and *Exact_Split* with 5 CPs and aspiration criterion performed

	Exact_Split		*TS_Split with diversification*						*TS_Split with no diversification*					
Number of VMs	Cost	CPU (s)	Cost gaps (%)			CPU (s)			Cost gaps (%)			CPU (s)		
			Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
5	7.17	0.07	0	0	0	0.05	0.1	0.09	0	0	0	0	0	0
10	17.25	0.48	0	0	0	0.08	0.15	0.14	0	0	0	0	0	0
15	23.58	20.9	0	0	0	0.23	0.31	0.3	0	0.94	0.18	0	0	0
20	35.13	228.8	0	0	0	0.47	0.59	0.56	0	1.13	0.71	0	0.01	0
25	47.27	405.1	0	0	0	0.93	1.08	1.06	0	2.68	1.25	0.01	0.01	0.01
30	37.88	711.9	0	0	0	1.49	1.73	1.62	0	3.5	1.56	0.01	0.01	0.01
35	58.54	1635.3	0	0	0	2.92	3.15	3.03	0	5.36	2.12	0.02	0.04	0.02
40	75.1	3147.6	0	0	0	4.95	5.14	5.07	0.05	4.65	2.08	0.03	0.04	0.03
45	62.17	3862.2	0	0	0	6.86	7.08	7.04	1.16	8.55	3.47	0.04	0.1	0.05
50	61.73	4134.3	0	0.64	0.18	10.09	10.64	10.36	2.21	10.9	5.49	0.06	0.09	0.07
55	87.69	4400.6	0	1.11	0.24	14.48	14.96	14.74	3.61	14.07	7.74	0.08	0.13	0.1
60	86.94	5801.9	0.01	0.81	0.21	19.96	20.12	20.4	4.23	19.89	9.52	0.11	0.16	0.12
65	91.89	5347.8	0.06	1.74	0.5	25.95	26.27	26.03	10.31	22.45	11.66	0.13	0.16	0.14
70	76.2	7846.1	0.1	1.95	1.17	35.17	35.41	35.39	9.74	25.02	11.92	0.17	0.29	0.2

Table 5.3 Comparison between *TS_Split* and *Exact_Split* with 5 CPs and no aspiration criterion performed

	Exact_Split		*TS_Split with diversification*						*TS_Split with no diversification*					
Number of VMs	Cost	CPU (s)	Cost gaps (%)			CPU (s)			Cost gaps (%)			CPU (s)		
			Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
5	7.17	0.07	0	0	0	0.03	0.1	0.07	0	0	0	0	0	0
10	17.25	0.48	0	0	0	0.08	0.13	0.11	0	0	0	0	0	0
15	23.58	20.9	0	0	0	0.18	0.3	0.25	0	1.07	0.29	0	0	0
20	35.13	228.8	0	0	0	0.36	0.58	0.44	0	1.43	0.72	0	0.01	0
25	47.27	405.1	0	0	0	0.7	1.12	0.78	0	2.83	1.78	0	0.01	0.01
30	37.88	711.9	0	0	0	1.11	1.45	1.16	0.01	3.73	2.26	0.01	0.01	0.01
35	58.54	1635.3	0	0	0	2.14	2.2	2.15	0.03	5.88	2.92	0.01	0.02	0.02
40	75.1	3147.6	0	0.04	0.01	3.84	4.19	4.01	0.05	6.02	3.25	0.02	0.03	0.03
45	62.17	3862.2	0	0.03	0.02	6.37	6.91	6.28	1.17	9.43	4.1	0.03	0.06	0.04
50	61.73	4134.33	0	0.63	0.34	9.66	9.95	9.67	2.34	12.01	5.56	0.04	0.08	0.05
55	87.69	4400.6	0.01	1.16	0.9	13.65	13.87	13.68	3.61	15.7	7.9	0.07	0.14	0.1
60	86.94	5801.9	0.01	0.9	0.82	18.87	19.12	18.41	4.29	28.01	11.61	0.05	0.2	0.12
65	91.89	5347.8	0.06	1.92	0.64	24.14	24.81	24.62	10.31	24.3	13.49	0.11	0.18	0.13
70	76.2	7846.01	0.18	2.69	1.55	34.49	34.88	34.69	9.76	27.1	14.05	0.17	0.21	0.18

Table 5.4 Comparison between *TS_Split* and *Exact_Split* with 10 CPs and aspiration criterion performed

	Exact_Split		*TS_Split with diversification*						*TS_Split with no diversification*					
Number of VMs	Cost	CPU (s)	Cost gaps (%)			CPU (s)			Cost gaps (%)			CPU (s)		
			Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
5	8.13	5.6	0	0.01	0	0.3	0.58	0.34	0	0.3	0.11	0	0	0
10	13.22	56.3	0	0	0	0.44	0.81	0.47	0	0.73	0.38	0	0.01	0.01
15	22.79	364.7	0	0	0	0.72	1.04	0.75	0	1.35	0.4	0.01	0.05	0.03
20	24.9	898.2	0	0	0	1.12	1.47	1.26	0	2.08	1.03	0.03	0.09	0.06
25	32.3	2000	0	0.06	0	2.23	2.75	2.34	0	3.9	1.35	0.06	0.13	0.07
30	36.05	2068.1	0	0.08	0	3.93	4.31	4.06	0.04	3.96	1.49	0.09	0.16	0.1
35	43.79	3722.2	0	0.12	0	6.62	7.05	6.71	0.12	4.35	2.97	0.08	0.17	0.11
40	50.51	5382.2	0	0.23	0.06	11.53	11.95	11.68	0.16	7.47	2.99	0.09	0.19	0.1
45	55.76	8703.1	0.01	0.34	0.15	17.3	17.86	17.55	1.08	10.09	5.83	0.11	0.25	0.13
50	70.87	12705.5	0	1.65	0.37	22.07	22.63	22.39	1.47	11.29	6.94	0.2	0.23	0.19
55	78.89	19159	0.01	1.9	0.78	34.21	35.12	34.62	2.19	16.85	9.07	0.22	0.31	0.3
60	123.76	19470.5	0.12	2.3	2.64	44.03	45.23	44.5	5.55	22.06	11.58	0.22	0.41	0.39
65	108.9	24818	0.22	3.15	2.5	57.98	58.44	58.09	9.65	23.35	14.92	0.29	0.57	0.4
70	101.65	49743.2	0.44	4.13	2.97	78.29	78.96	78.83	18.04	32.1	19.64	0.4	0.8	0.53

Table 5.5 Comparison between *TS_Split* and *Exact_Split* with 10 CPs and no aspiration criterion performed

	Exact_Split		*TS_Split with diversification*						*TS_Split with no diversification*					
Number of VMs	Cost	CPU (s)	Cost gaps (%)			CPU (s)			Cost gaps (%)			CPU (s)		
			Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
5	8.13	5.6	0	0	0	0.16	0.57	0.25	0	0.31	0.12	0	0	0
10	13.22	56.3	0	0	0	0.33	0.68	0.36	0	0.75	0.41	0	0.01	0.01
15	22.79	364.7	0	0.24	0.07	0.54	0.96	0.67	0.04	1.49	0.4 7	0.01	0.03	0.02
20	24.9	898.2	0	0.21	0.09	1.15	1.39	1.18	0	2	1.03	0.01	0.07	0.04
25	32.3	2000	0	0.25	0.12	1.99	2.34	2.05	0	4.13	1.41	0.02	0.1	0.05
30	36.05	2068.1	0	0.47	0.16	3.2	3.51	3.19	0.05	4.01	1.5	0.02	0.12	0.07
35	43.79	3722.2	0	0.43	0.24	5.19	6.16	5.21	0.11	5.38	3.01	0.05	0.15	0.1
40	50.51	5382.2	0.01	0.31	0.18	9.34	10.12	9.54	0.18	8.85	3.57	0.08	0.16	0.09
45	55.76	8703.1	0.01	1.48	0.27	14.23	14.72	14.41	1.08	13.16	6.13	0.1	0.17	0.11
50	70.87	12705.5	0.06	3.76	0.58	18.24	19.47	18.58	1.89	12.48	8.15	0.18	0.2	0.14
55	78.89	19159	0.02	4.12	1.19	28.61	29.43	28.77	2.34	17.2	9.57	0.2	0.25	0.21
60	123.76	19470.5	0.43	5.01	2.9	36.71	38.68	37.07	5.57	23.18	11.97	0.19	0.33	0.24
65	108.9	24818	0.56	5.83	3.08	48.06	49.14	48.59	9.74	25.08	15.83	0.34	0.41	0.38
70	101.65	49743.2	0.5	6.91	4.84	68.45	69.36	69.85	18.67	34.3	20.6	0.3	0.45	0.42

In Table 5.3, which presents the results of comparison with 5 CPs with no aspiration criterion performed, we can observe on average a slight deterioration in the quality of the solutions, with a maximal cost gap up to 2.69% when the long-term memory is executed. With no *Diversification* mechanism, the cost gap is more significant without the aspiration criterion, giving maximums of gap up to 28%.

Same conclusions are noticed in Table 5.4 and Table 5.5. *TS_Split* is as effective when the complexity of the problem is increased with 10 CPs. As shown in Table 5.4, with the long-term memory and the aspiration criterion performed, most of the cost gaps are bellow 0.4%, with an average of gap between 0% and 2.97%. There too, with no aspiration criterion, as shown in Table 5.5, the cost gap increases a little, giving an average of gap up to 4.04%. With no *Diversification* mechanism, the quality of solutions is considerably degraded, giving a maximal cost gap up to 32.1% and 34.3%, respectively with and without the aspiration criterion performed.

The analysis of the execution time shows that the CPU time is correlated and increases on average according to the size of the instances. With the exact method, as expected, the CPU time increases exponentially from 0.07 s with 5 VMs to 7846.1 s with 70 VMs in the scenario with 5 CPs, and from 5.6 s with 5 VMs to 49743.2 s with 70 VMs in the scenario with 10 CPs. The proposed heuristic significantly reduces the CPU time which becomes linear, besides being effective in finding near-optimal solutions. As shown in Table 5.2 and Table 5.4, with the long-term memory we can respectively observe CPU time averages of only 0.09 s up to 35.39 s with 5 CPs, and from 0.34 s up to 78.83 s with 10 CPs. With no aspiration criterion performed, Table 5.3 and Table 5.5 show CPU times a little more reduced, with averages ranging from 0.07 s to 34.69 s and from 0.25 s to 69.85 s, respectively with 5 CPs and 10 CPs. This is due to the fact that without aspiration criterion, in each iteration the heuristic only evaluates non-tabu moves, which is less expensive in terms of calculation time, however we have solutions slightly of lower quality. Regarding all the four tables, without the long-term memory, the CPU times are very low, with a maximum up to 0.29 s and 0.8 s respectively with 5 CPs and 10 CPs, but at the price of much less good quality of solutions.

Results obtained with *Scenario 2* are presented in Table 5.6, Table 5.7, Table 5.8 and Table 5.9. For such large sized instances of the problem, the *Diversification* mech/anism was necessary mostly because of the high number of VLs the heuristic has to satisfy. The simple algorithm without long-term memory can easily be trapped in bad local optima, sometimes leading to non-feasible solutions due to some violated VLs. In each of these four tables, for each instance, we report first the minimal, maximal and mean evaluation cost. Then, we present the gap between the mean cost and the maximal cost found by the heuristic.

Table 5.6 *TS_Split* results for large sized VNRs with 5 CPs and aspiration criterion performed

TS_Split with diversification									
Number of VMs	Cost Min	Cost Max	Cost Mean	Mean gap to max cost (%)	# nbIter (mean)	# bestIter (mean)	CPU (s) Min	CPU (s) Max	CPU (s) Mean
75	151.02	151.34	151.15	0.13	5750	1273	35.42	37.66	36.02
100	200.45	201.75	201.67	0.04	5735	931	37.17	42.59	38.98
125	250.06	255.1	254.85	0.1	5946	1781	38.03	47.23	41.36
150	307.43	310.96	309.9	0.34	5592	174	42.3	49.55	44.41
175	360.39	362.89	362.08	0.22	5948	1600	46.36	52.86	48.37
200	359.15	361.05	360.83	0.06	6046	2291	50.27	56.32	52.43
225	406.35	408.31	407.92	0.1	6608	2058	51.04	61.03	58.26
250	439.77	440.8	440.32	0.11	6766	4766	60.12	71.74	66.44
275	474.16	475.14	474.62	0.1	6786	4166	81.07	102.24	92.06
300	491	494.06	493.4	0.13	6316	3034	107.33	119.47	112.6
325	504.26	507.85	506.9	0.19	6179	1163	136.84	141.42	139.11
350	555.16	557.58	556.74	0.15	5922	2021	159.14	178.64	163.82
375	539.18	541.98	540.03	0.36	6135	1606	202.35	210.04	206.82
400	589.2	592.24	591.08	0.2	6315	1385	253.82	264.11	258.64
425	632.09	636.2	635.13	0.17	6560	1482	319.07	359.09	334.21
450	597.94	602.18	602	0.03	6095	2074	360.74	405.41	371.91
475	649	652.91	650.88	0.31	6285	1738	458.19	470.31	464.14
500	688.23	691.74	690.17	0.23	6330	1381	568.32	633.94	608.36

Table 5.7 *TS_Split* results for large sized VNRs with 5 CPs and no aspiration criterion performed

TS_Split with diversification									
Number of VMs	Cost Min	Cost Max	Cost Mean	Mean gap to max cost (%)	# nbIter (mean)	# bestIter (mean)	CPU (s) Min	CPU (s) Max	CPU (s) Mean
75	149.84	151.3	150.85	0.32	5984	3618	34.64	37.12	35.82
100	198.59	201.75	200.05	0.84	5692	1900	35.1	43.18	37.58
125	250	255.1	252.5	1.02	5985	3514	37.83	46.9	40.96
150	305.2	310.96	307.8	1.02	5592	174	42.01	49.11	43.21
175	360.39	362.63	360.41	0.68	5948	1600	46.36	52.86	48.37
200	354.15	361.05	357.91	0.87	6046	2291	50.27	56.32	52.43
225	400.66	408.07	405.34	0.73	6608	2058	50.94	60.13	57.62
250	421.41	439.68	428.83	2.72	6824	3824	60.07	70.87	64.58
275	455.16	475.02	466.88	1.73	6368	2176	80.76	92.42	84.96
300	491.03	494.06	493.27	0.16	6187	2030	103.89	117.27	106.96
325	500.12	507.14	505.39	0.48	6252	1578	133.52	141.27	136.85
350	551.37	555.95	555.88	0.3	5915	1424	155.24	166.68	159.92
375	530.99	541.98	537.2	0.88	6153	2481	199.69	206.11	203.08
400	583.24	589.07	585.39	1.16	6409	1605	248.25	276.69	257.85
425	603.82	636.2	629.72	1.02	6666	1784	317.29	336.35	328.09
450	575.75	602.18	598.43	0.62	6252	2902	349.69	379.97	369.18
475	628.49	652.21	643.76	1.4	6144	1872	439.75	450.44	444.03
500	675.4	691.74	687.95	0.55	6857	4170	552.19	579.69	574.99

Table 5.8 *TS_Split* results for large sized VNRs with 10 CPs and aspiration criterion

TS_Split with diversification									
Number of VMs	Cost Min	Cost Max	Cost Mean	Mean gap to max cost (%)	# nbIter (mean)	# bestIter (mean)	CPU (s) Min	CPU (s) Max	CPU (s) Mean
75	102.92	102.95	102.94	0.01	11321	4587	84.37	85.4	84.69
100	139.92	139.92	139.92	0	11133	6037	89.57	90.25	89.89
125	169.35	170.11	170.03	0.05	11475	1175	89.79	91.13	90.56
150	204.82	204.82	204.82	0	11515	1559	96.2	98.42	97.61
175	242.02	242.97	242.39	0.24	11295	3139	98.38	99.95	99.18
200	275.94	277.94	276.16	0.64	11461	7164	99.25	102.7	100.58
225	301.33	306.54	305.6	0.31	11540	3668	116.08	124.39	120.41
250	325.26	336.25	334.73	0.45	12098	5311	176.42	187.29	179.89
275	375.65	380.05	379.09	0.25	11676	1302	216.1	230.7	221.04
300	418.76	436.2	434.87	0.3	12281	3960	304.06	327.79	311.32
325	441.97	444.16	444.02	0.03	12866	5411	383.57	428.49	405.36
350	450.43	452.95	452.43	0.11	11954	2256	446.87	513.39	461.05
375	506.57	507.78	507.41	0.07	12210	4705	572.15	640.47	594.39
400	515.79	518.4	516.83	0.3	12987	4583	717.89	803.43	766.97
425	600.13	601.97	601.94	0	12482	3641	868.12	961.85	897.87
450	643.49	646.28	646.01	0.04	12539	3249	1058.32	1198.71	1109
475	653.28	676.34	674.29	0.3	12427	5999	1327.05	1404.47	1369.52
500	710.62	711.98	711.82	0.02	12865	8465	1693.68	1848.32	1765.91

Table 5.9 *TS_Split* results for large sized VNRs with 10 CPs and no aspiration criterion

TS_Split with diversification									
Number of VMs	Cost Min	Cost Max	Cost Mean	Mean gap to max cost (%)	# nbIter (mean)	# bestIter (mean)	CPU (s) Min	CPU (s) Max	CPU (s) Mean
75	100.48	102.95	101.86	1.06	11321	4587	72.68	74.3	72.34
100	139.07	139.92	139.73	0.14	11133	6037	76.94	78.63	77.37
125	163.27	170.11	167.76	1.38	11475	1175	79.25	83.07	82.67
150	179.75	204.82	201.69	1.53	11515	1559	88.57	90.71	90.54
175	240.81	242.18	241.93	0.43	11295	3139	93.09	94.25	94.53
200	271.02	277.94	275.92	0.73	11461	7164	96.12	99.59	97.83
225	300.87	304.46	303.68	0.93	12006	7651	113.64	121.36	118.16
250	321.6	336.25	327.38	2.64	12365	7063	166.5	178.29	173.22
275	372.28	379.74	375.43	1.22	11758	3755	202.86	224.8	210.06
300	407.25	436.2	426.54	2.21	12682	7419	291.54	330.06	305.35
325	433.2	442.36	441.65	0.57	13406	6152	359.21	428.42	399.82
350	450.01	450.13	450.08	0.63	12462	6053	435.19	500.46	472.4
375	500.38	507.78	502.19	1.1	13789	6861	559.1	623.49	589.42
400	504.72	515.89	511.75	1.28	12497	4451	644.94	749.72	669.28
425	576.52	600.74	596.86	0.85	12845	7225	853.69	931.81	979
450	635.25	646.28	643.48	0.43	12505	3025	995.55	1113.31	1069.45
475	644.05	676.28	665.16	1.65	11976	8080	1200.96	1273.03	1230.36
500	696.86	710.12	702.08	1.39	12678	6678	1450.24	1778.22	1591.96

Also, among the 10 simulations that could reach the best solution found by the heuristic, we report the average of the total number of iterations *nbIter* that was necessary, as well as the average of the iterations *bestIter* where the best solution was found. The minimal, maximal and mean execution time is also given. As shown in Table 5.6 and Table 5.8, the algorithm is effective even for large sized instances, giving averages of solution cost very close to the maximal values found. Indeed, the average gap for all instances between the mean and the maximal cost is about 0.16% and 0.17%, respectively with 5 CPs and 10 CPs. In addition, the mean *bestIter* observed, regarding the total number *nbIter*, shows that on average the algorithm can converge fast towards the best solution found. On the other hand, as expected, the execution time is higher with the large cases of the problem, but still satisfactory given the complexity of the scenarios that makes harder to effectively explore all the solution space. With 5 CPs, we can note averages of CPU time ranging from 36.02 s for 75 VMs to 608.36 s for 500 VMs. With 10 CPs, the mean CPU time increases naturally from 84.69 s to 1765.91 s, respectively for instances from 75 VMs to 500 VMs.

Results obtained in Table 5.7 and Table 5.9 confirm that without the aspiration criterion, less good qualities of solution are noticed, with an average gap for all instances between the mean and the maximal cost of about 0.92% and 1.12%, respectively with 5 CPs and 10 CPs. Furthermore, by observing the mean *bestIter*, without the aspiration criterion it seems harder for the algorithm to converge more easily towards the best solution found. However, the execution time is reduced, giving on average from 35.82 s to 574.99 s with 5 CPs, and from 72.34 s to 1591.96 s with 10 CPs.

The results analysis resulting from the *Experiment 1* shows that the proposed algorithm is effective in solving such a combinatorial networking problem, even without the aspiration criterion performed, by offering a good tradeoff between the quality of the solutions and the highly reduced computing time.

5.6.3.2 *TS_Split* with Experiment 2

The performance of the proposed VNRs splitting strategy is first evaluated with the acceptance rate, which represents one of the most conclusive metrics to evaluate the efficiency of a successful performance-based VNRs splitting strategy. The acceptance rate is measured as the number of VNRs that are successfully mapped by all CPs selected by the splitting strategy, divided by the total number of incoming VNRs. In our experiment, if only one VM or one VL of a request has a failed embedding (including the failed intercloud connections), we consider the entire VNR rejected.

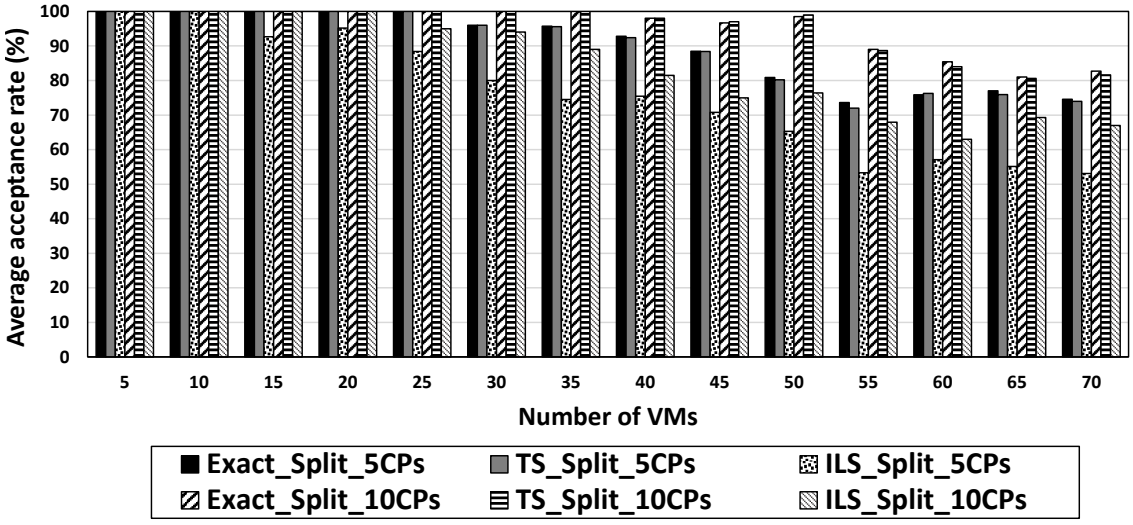

Figure 5.5 Average acceptance rate for small sized VNRs

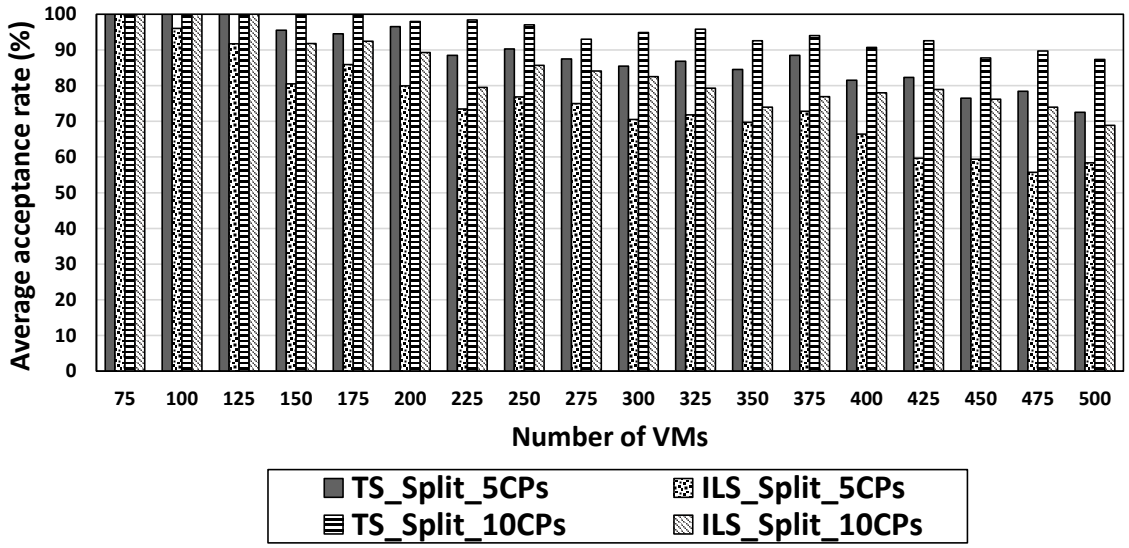

Figure 5.6 Average acceptance rate for large sized VNRs

From Figure 5.5, which presents the average acceptance rate per size of request for small sized VNRs (i.e. with *Scenario 3*), it can be seen that the acceptance rate decreases on average according to the number of communicating VMs. This is primarily due to the fact that available resources of CPs can be limited over time because of requests that have not yet expired. Thus, on average it becomes harder to satisfy all the QoS constraints for VNRs with a greater number of VMs. Our splitting strategy, with both the exact and the proposed heuristic approaches, leads to higher acceptance rates than the approach of (Leivadeas *et al.*,

2013). Indeed, the mean acceptance rate considering all requests is about 89.64% with 5 CPs and 95.09% with 10 CPs with *Exact_Split*, and about 89.34% with 5 CPs and 94.92% with 10 CPs with *TS_Split*. The latter leads to a little more requests rejections than the exact method, but almost negligible, which proves that our heuristic is as effective as the exact method to perform good decisions in the splitting strategy. However, *ILS_Split* approach is more subject to VNRs rejections, even with the large instances of the problem. For small sized VNRs as depicted in Figure 5.5, *ILS_Split* gives a mean acceptance rate for all requests of about 75.78% with 5 CPs and 84.15% with 10 CPs.

For large sized VNRs with *Scenario 4*, as illustrated in Figure 5.6, *TS_Split* gives a mean acceptance rate for all requests of about 88.29% with 5 CPs and 95.1% with 10 CPs, while with *ILS_Split* we note 74.65% of mean acceptance rate with 5 CPs and about 83.97% with 10 CPs. This means that with *TS_Split* strategy, we improve on average the acceptance rate by approximately 15.34% and 15.76%, respectively for small and large sized VNRs. This can be explained by the fact that *ILS_Split* strategy tends to load most of the resource demands to CPs having the largest numbers of data centers and substrate links. This may not be a representative performance criterion if data centers are heterogeneous with different capacities. Also, the bandwidth capacities on substrate paths are not necessarily related to the number of substrate links they consider in the link provisioning cost, which can results in more rejected VLs.

Table 5.10 Average computing time for small sized VNRs

	CPU (s) with 5 CPs			CPU (s) with 10 CPs		
Number of VMs	*Exact_Split*	*TS_Split*	*ILS_Split*	*Exact_Split*	*TS_Split*	*ILS_Split*
5	0.09	0.14	0.11	5.2	0.4	2.04
10	0.9	0.19	0.18	63.76	0.69	4.04
15	40.93	0.32	0.59	438.39	0.94	4.92
20	200.37	0.6	0.67	885.77	1.13	6.89
25	394.7	1.18	0.85	1664.68	2.79	10.04
30	912.5	1.7	2.12	2485.97	4.35	12.48
35	1836.31	3.45	4.61	3811.83	7.37	15.13
40	3043.93	5.43	6.68	6307.88	12.01	18.04
45	3688.16	7.15	10.24	7746.59	17.24	25.05
50	4123.48	11.01	19.36	12326.89	23.8	36.73
55	4375.38	15.05	27.37	19487.75	34.98	57.15
60	5741.5	20.41	33.45	21795	46.51	60.54
65	6947.85	26.8	37.1	29121.24	59.97	70.1
70	7486.48	35.65	51.52	49672.83	79.89	114.39

Table 5.11 Average computing time for large sized VNRs

	CPU (s) with 5 CPs		CPU (s) with 10 CPs	
Number of VMs	*TS_Split*	*ILS_Split*	*TS_Split*	*ILS_Split*
75	35.5	55.1	84.7	159.8
100	37.1	75.3	89.9	218.4
125	43.6	90.3	90.6	261.9
150	52.4	118.8	97.6	344.5
175	63	134.6	109.2	390.3
200	71.5	149.2	119.6	432.7
225	77.8	163.9	120.4	475.3
250	85.2	175.7	179.9	509.5
275	95.8	198.1	221	574.5
300	133.9	225.4	311.3	653.7
325	140.7	238	405.4	690.2
350	162.9	274.9	461.1	797.2
375	205.6	301.7	594.4	874.9
400	255	347.5	767	1007.8
425	328.3	422	897.9	1223.8
450	400	512.1	1109	1485.1
475	467.3	643.4	1369.5	1865.9
500	591.6	704.4	1765.9	2042.8

The execution time results are presented in Table 5.10 and Table 5.11, respectively for *Scenario 3* and *Scenario 4*. For small sized VNRs in Table 5.10, the execution time with *Exact_Split* is naturally exponential, as the solver CPLEX uses *Branch-and-Bound* methods. Considering all instances, in comparison with *TS_Split* which gives an average of about 9.22 s and 20.86 s respectively with 5 CPs and 10 CPs, *ILS_Split* takes a little more time to compute the splitting phase, resulting in an average of about 13.92 s and 33.4 s respectively with 5 CPs and 10 CPs. Same results can be noticed in Table 5.11 with large sized VNRs. The CPU time with *TS_Split* is on average 180.4 s and 488.57 s respectively with 5 CPs and 10 CPs, while with *ILS_Split* it is about 268.35s and 778.23 s respectively with 5 CPs and 10 CPs. This is probably due to the large number of iterations they assess for the LS, and the high perturbation strength of 80% they perform at each iteration for such a combinatorial problem, leading the LS to iteratively calculate almost all the objective function.

We also evaluate the average splitting rate, calculated as the number of assigned segments to the selected CPs at each incoming VNR, divided by the total number of CPs. As shown in Figure 5.7 and Figure 5.8, the splitting rate increases on average according to the size of requests. For small sized VNRs as depicted in Figure 5.7, both *Exact_Split* and *TS_Split* approaches give substantially same results, with an average splitting rate for all requests of about 51.82% and 36.73% respectively with 5 CPs and 10 CPs, while *ILS_Split* gives much

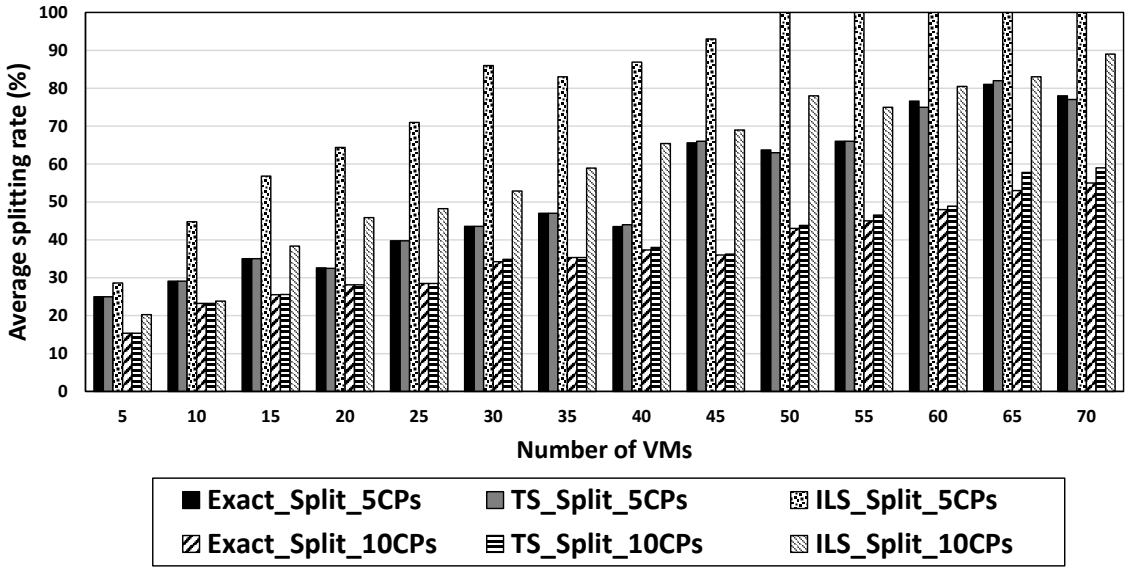

Figure 5.7 Average splitting rate for small sized VNRs

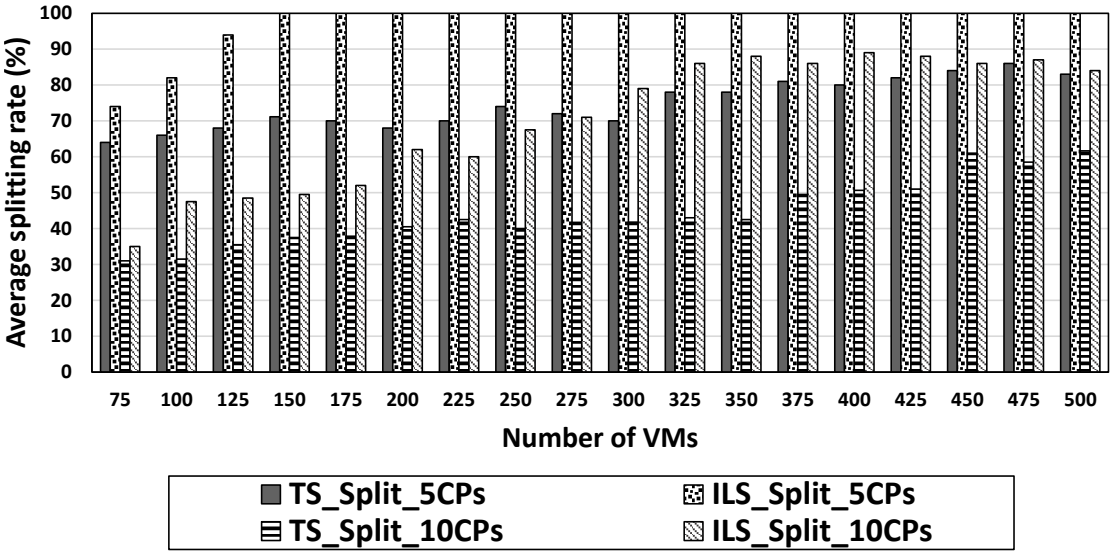

Figure 5.8 Average splitting rate for large sized VNRs

higher splitting rates, which are on average 79.65% and 59.16% respectively with 5 CPs and 10 CPs. For large sized VNRs as depicted in Figure 5.8, the average splitting rate for all requests with *TS_Split* is about 74.73% and 51.49% respectively with 5 CPs and 10 CPs, while with *ILS_Split* it is about 97.22% and 70.83% respectively with 5 CPs and 10 CPs. This means that our strategy efficiently assigns most of the VNRs to 3 up to 5 CPs among the participating CPs, thus avoiding useless intercloud traffics. However, *ILS_Split* assigns

most of the VNRs on average to 4 up to 7 CPs, without giving better acceptance rates. A VNR splitting rate unnecessarily high should be avoidable. It generally introduces a large number of VLs transiting through the intercloud paths, resulting in high overall delays and huge additional expenditures for the SP.

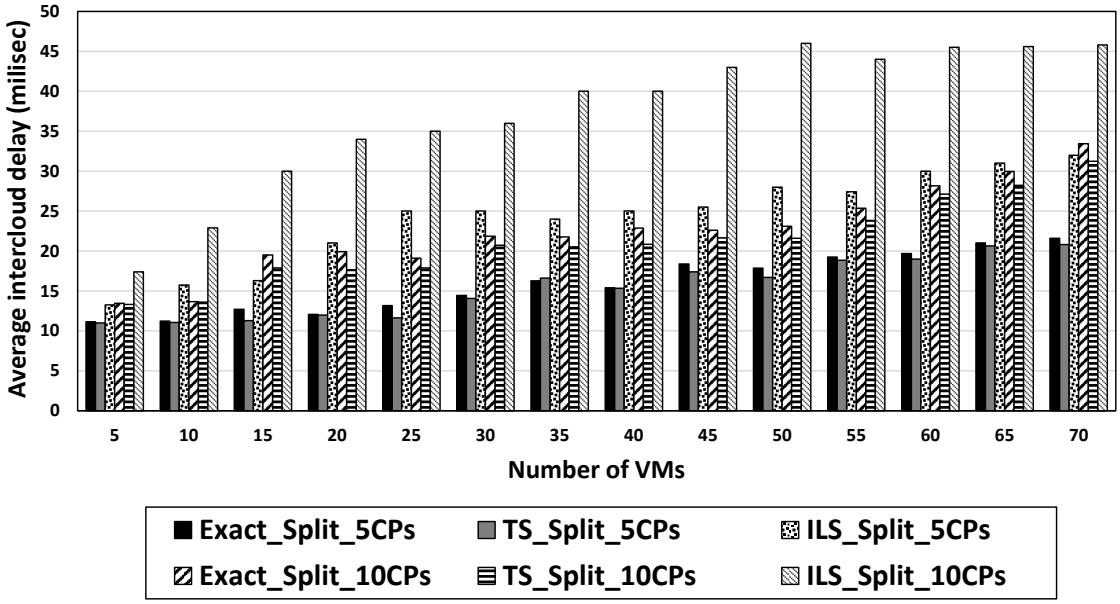

Figure 5.9 Average intercloud delay for small sized VNRs

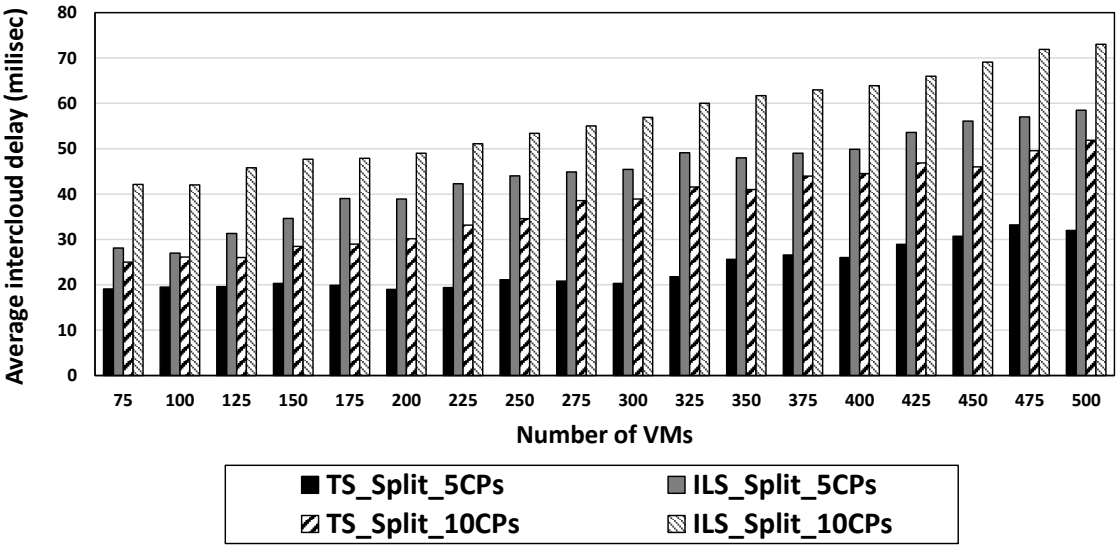

Figure 5.10 Average intercloud delay for large sized VNRs

Finally, we present the average intercloud delay in Figure 5.9 and Figure 5.10, which is very closely in correlation with the longer of chosen paths and the splitting rate. The delay

represents the performance metric that influences the most the QoS. As expected, for both scenarios the intercloud delay increases on average according to the size of requests. In Figure 5.9, we can see that *TS_Split* gives less intercloud delays than *Exact_Split* and *ILS_Split*. In comparison with the exact method, this is due to the fact that *TS_Split* algorithm is able to perform a VLs routing by considering not only the shortest path (defined by the number of hops), but also the delay on the path. The solver of exact method meanwhile randomly chooses a path between those with the same shortest length, regardless of the delay on the paths. *ILS_Split* approach gives more intercloud delays considering all requests, which are on average about 21.87 ms with 5 CPs and 34.93 ms with 10 CPs, against 15.45 ms with 5 CPs and 21.15 ms with 10 CPs for *TS_Split* algorithm.

For large sized VNRs, Figure 5.10 depicts the same conclusion, with an average intercloud delay of about 44.27 ms with 5 CPs and 56.64 ms with 10 CPs for *ILS_Split* approach, while *TS_Split* results in an average delay of about 23.54 ms with 5 CPs and 37.51 ms with 10 CPs. This means that on average the intercloud delay is improved by about 40.29% with our *TS_Split* approach. We can also notice that a high splitting rate has a significant influence on the intercloud delay, by resulting in more VMs communicating through intercloud links. Thus, we can conclude that our approach penalizes more effectively long intercloud paths than *ILS_Split* approach.

5.7 Conclusion

In this paper, the NP-hard problem of VNE across multiple IaaS-based cloud networks has been addressed. A TS approach including a long-term memory mechanism has been proposed in order to perform, in polynomial time, an efficient VNRs splitting strategy formalized as a mathematical ILP model. The proposed strategy aims at improving the performance and the QoS of resulting VNR segments that are mapped onto the selected cloud infrastructure networks. The comparison results with the exact method, used to execute small sized instances of the problem, show that the proposed algorithm is able to generate, in a highly reduced computing time, solution costs very close to the upper bounds, with an average cost gap from about 0% to a maximum of 2.97%.

Simulations performed with large instances of the problem and comparing our approach with an other baseline method, show the efficiency of the proposed approach in dealing with the scalability aspect of such a combinatorial networking problem. Our approach improves several performance criteria, including the acceptance rate and the network communication delay, which are respectively improved of about 15.55% and 40.29%.

Future works might be interested in developing efficient and fast dynamic hybrid metaheuristics, by combining different approaches and optimization techniques to solve large instances of the problem. Furthermore, as the SP is also interested in minimizing the resource provisioning price, it is worthwhile to consider in future works a multi-objective optimization approach, which can extend the proposed mathematical model by also taking into account the minimization of the SP expenditure in the splitting strategy.

Appendix A FORMULATION OF THE VNR SEGMENTS INTRACLOUD MAPPING PROBLEM

In this appendix, we present the MILP formulation of the adopted intra-cloud mapping approach, which follows the work of (Larumbe et Sanso, 2013). Details related to the setup of cost parameters used for our simulations are also given. All the notation used for the modeling of the intra-cloud substrate network and the VNR segments, as well as the formulation of the problem, can be consulted in Table 5.12.

Table 5.12 Notation for the VNR segments intracloud mapping problem

Symbols	Description
	Global sets
A	Set of node categories
T	Set of link types
R	Set of computational resource types
I	Set of CPs
	Virtual network request segment
G_i^V	Graph representing the VNR segment assigned to CP $i \in I$
N_i^V	Set of VMs assigned to CP $i \in I$
L_i^V	Set of virtual links assigned to CP $i \in I$
N_{ai}^V	Set of VMs of node category $a \in A$ assigned to CP $i \in I$
L_{ti}^V	Set of virtual links of type $t \in T$ exclusively assigned to CP $i \in I$
$L_{t\phi_i}^V$	Set of virtual links of type $t \in T$ assigned to an inter-cloud link of endpoint CP $i \in I$
q_{rv}	Amount of resource $r \in R$ required by VM $v \in N_i^V$
b_l	Bandwidth demand of virtual link $l \in L_i^V$
δ_l	Maximum delay allowed for virtual link $l \in L_i^V$

Table 5.12 Notation for the VNR segments intracloud mapping problem

	Substrate intracloud network
G_i^S	Graph representing the substrate network of CP $i \in I$
N_i^S	Set of substrate nodes in G_i^S
L_i^S	Set of substrate links in G_i^S
D_i	Set of DCs of CP $i \in I$
ϕ_i	PoP node (transit network) of CP $i \in I$
$\mathcal{F}_i$	Set of all paths in CP i's network
$\mathcal{P}_{dc}$	Set of all paths between DCs d and c, $d, c \in D_i$
$\mathcal{P}_{d\phi_i}$	Set of all paths between DC $d \in D_i$ and PoP node ϕ_i
$\mathcal{K}_e$	Set of all paths in $\mathcal{F}_i$ spanning link $e \in L_i^S$
B_{te}	Available bandwidth capacity of the channel of type $t \in T$ of link $e \in L_i^S$
d_e	Delay bound on link $e \in L_i^S$
Q_{rad}	Available capacity of resource $r \in R$ in DC $d \in D_i$for nodes of category $a \in A$ ($Q_{rad} \in \mathbb{N}_1$)
U_{adr}	Usage of resource $r \in R$ by all VMs of node category $a \in A$ assigned to DC $d \in D_i$
E_{adr}	Average power (in watts) consumed by a VM of node category $a \in A$ assigned to DC $d \in D_i$ in terms of resource $r \in R$
ω_d	Average power (in watts) consumed by all VMs assigned to DC $d \in D_i$
ρ_d	Power Usage Effectiveness (PUE) of DC $d \in D_i$
θ_d	CO_2 emissions in DC $d \in D_i$ (in g/KWh)
	Costs
$c_{rai}^{\mathcal{S}}$	Unit resource cost for CP $i \in I$ for using resource $r \in R$ for nodes of category $a \in A$ (in \$/unit)
c_{ti}	Unit bandwidth cost for CP $i \in I$ for using a link channel of type $t \in T$ (in \$/Mbps)
$c_{ti}^{\mathcal{E}}$	Extra unit bandwidth cost for CP $i \in I$ for using a link channel of type $t \in T$ (in \$/Mbps)
c_d^{ω}	Unit electricity cost in DC $d \in D_i$ (in \$/MWh)
$\wp^{\mathcal{D}}$	Penalty for each millisecond of delay on virtual links (in \$/ms)
$\wp_d^{\mathcal{O}}$	Penalty for emitting CO_2 in DC $d \in D_i$ (in \$/tonne)
$C_i^{\mathcal{S}}$	Total computing resource cost for CP $i \in I$ (in \$/h)
$C_i^{\mathcal{T}}$	Total traffic cost for CP $i \in I$ (in \$/h)
$C_i^{\mathcal{D}}$	Total delay penalty for CP $i \in I$ (in \$/h)
C_i^{ω}	Total energy cost for CP $i \in I$ (in \$/h)
$C_i^{\mathcal{O}}$	Total environmental penalty for CP $i \in I$ (in \$/h)
	Decision variables
X_{vn}	Binary variable set to **1** if VM $v \in N_{ai}^V$, $a \in A$, is assigned to substrate node $n \in N_i^S$; **0** otherwise
$Y_{l\varphi}$	Binary variable set to **1** if virtual link $l \in L_{ti}^V$, $t \in T$, is assigned to path $\varphi \in \mathcal{P}_{dc}$, $d, c \in D_i$; **0** otherwise
$Z_{k\gamma}$	Binary variable set to **1** if virtual link $k \in L_{t\phi_i}^V$, $t \in T$, is assigned to path $\gamma \in \mathcal{P}_{d\phi_i}$, $d \in D_i$; **0** otherwise

A.1 Adopted intracloud mapping approach

The following are the formulas that define each cost and penalty of the multicriteria objective function. In our solution, the mapping of a VM remains at the data center level. VMs requirements are expressed in terms of computational resource demand in set R, not in terms of number of servers. Therefore, some formulas originally defined in (Larumbe et Sanso, 2013) were re-stated. Moreover, we did not consider the path splitting scenario, neither the data centers CAPEX and OPEX since we assign VMs to existing data centers. Note that each cost is defined in dollars per hour ($/h).

Servers cost is re-stated as computational resources cost and is calculated as follows:

$$C_i^{\mathcal{S}} = \sum_{d \in D_i} \sum_{r \in R} \sum_{a \in A} \sum_{v \in N_{ai}^V} c_{rai}^{\mathcal{S}} q_{rv} X_{vd} \tag{A.1}$$

The total traffic cost for CP i is defined as follows, with an extra cost stated for intercloud virtual links:

$$\begin{aligned} C_i^{\mathcal{T}} = & \sum_{d \in D_i} \sum_{c \in D_i} \sum_{\varphi \in \mathcal{P}_{dc}} \sum_{e \in \varphi} \sum_{t \in T} \sum_{l \in L_{ti}^V} c_{ti} b_l Y_{l\varphi} \\ & + \sum_{d \in D_i} \sum_{\gamma \in \mathcal{P}_{d\phi_i}} \sum_{e \in \gamma} \sum_{t \in T} \sum_{k \in L_{t\phi_i}^V} \left(c_{ti} + c_{ti}^{\mathcal{E}} \right) b_k Z_{k\gamma} \end{aligned} \tag{A.2}$$

The total delay penalty for CP i is re-defined as follows:

$$C_i^{\mathcal{D}} = \wp^{\mathcal{D}} \left(\begin{array}{l} \sum_{d \in D_i} \sum_{c \in D_i} \sum_{\varphi \in \mathcal{P}_{dc}} \sum_{e \in \varphi} \sum_{t \in T} \sum_{l \in L_{ti}^V} d_e Y_{l\varphi} \\ + \sum_{d \in D_i} \sum_{\gamma \in \mathcal{P}_{d\phi_i}} \sum_{e \in \gamma} \sum_{t \in T} \sum_{k \in L_{t\phi_i}^V} d_e Z_{k\gamma} \end{array} \right) \tag{A.3}$$

The average power consumed by all VMs assigned to data center $d \in D_i$ is re-stated as follows, estimated as a function of the utilization of CPU resource and disk resource, by following the principles used in (Justafort *et al.*, 2015):

$$\omega_d = \sum_{a \in A} \left(E_{adr_1} U_{adr_1} + E_{adr_2} U_{adr_2} \right), \quad \forall\, d \in D_i, \tag{A.4}$$

where E_{adr_1} and E_{adr_2} represent (in watts) the average power consumed by a VM of node category $a \in A$ assigned to data center d, respectively in terms of CPU resource and disk resource. U_{adr_1} and U_{adr_2} define respectively the CPU resource usage and the disk resource usage by all VMs of node category $a \in A$ assigned to data center d, given by:

$$U_{adr} = \frac{\sum\limits_{v \in N_{ai}^V} q_{rv} X_{vd}}{\max\left(Q_{rad}, 1\right)}, \qquad \forall\, d \in D_i,\ a \in A,\ r \in R \tag{A.5}$$

The total energy cost for CP i is then defined as follows:

$$C_i^{\omega} = 10^{-6} \sum_{d \in D_i} c_d^{\omega} \rho_d \omega_d, \tag{A.6}$$

The total environmental penalty for CP i is given as follows:

$$C_i^{\mathcal{O}} = 10^{-9} \sum_{d \in D_i} \wp_d^{\mathcal{O}} \theta_d \rho_d \omega_d \tag{A.7}$$

The objective function for the intracloud mapping phase is then defined as follows, with each cost weighted by a parameter that allows CPs to modify the costs priority:

MIN

$$\alpha \mathcal{C}_i^{\mathcal{S}} + \beta \mathcal{C}_i^{\mathcal{T}} + \lambda \mathcal{C}_i^{\mathcal{D}} + \varpi \mathcal{C}_i^{\omega} + o \mathcal{C}_i^{\mathcal{O}} \tag{A.8}$$

Subject to:

$$\sum_{n \in N_i^S \setminus D_i} X_{vn} = 0, \qquad \forall\, v \in N_{ai}^V,\ a \in A \tag{A.9}$$

$$\sum_{d \in D_i} X_{vd} = 1, \qquad \forall\, v \in N_{ai}^V,\ a \in A \tag{A.10}$$

Constraints (A.9) and (A.10) ensure respectively that VMs are only assigned to data centers, and each VM must be assigned to exactly one data center.

$$Y_{l\varphi} \leq \frac{X_{ud} + X_{vc}}{2}, \qquad \begin{array}{l} \forall\ l = uv \in L_{ti}^{V},\ t \in T, \\ u, v \in N_{ai}^{V} (v \neq u),\ a \in A, \\ \varphi \in \mathcal{P}_{dc},\ d, c \in D_i \end{array} \tag{A.11}$$

$$Z_{k\gamma} \leq X_{vd}, \qquad \begin{array}{l} \forall\ k = v\phi_i \in L_{t\phi_i}^{V},\ t \in T,\ v \in N_{ai}^{V}, \\ \gamma \in \mathcal{P}_{d\phi_i},\ d \in D_i \end{array} \tag{A.12}$$

Constraints (A.11) and (A.12) state respectively the binary value of variables $Y_{l\varphi}$ and $Z_{k\gamma}$.

$$\sum_{d \in D_i} \sum_{c \in D_i} \sum_{\varphi \in \mathcal{P}_{dc}} Y_{l\varphi} = 1, \qquad \forall\ l \in L_{ti}^{V},\ t \in T \tag{A.13}$$

$$\sum_{d \in D_i} \sum_{\gamma \in \mathcal{P}_{d\phi_i}} Z_{k\gamma} = 1, \qquad \forall\ k \in L_{t\phi_i}^{V},\ t \in T \tag{A.14}$$

Constraint (A.13) ensures that each virtual link exclusively assigned to a CP must be mapped to a unique path between two different data centers, otherwise to a unique path intra-data center. Constraint (A.14) ensures that each virtual link partially assigned to a CP must be mapped to a unique path between a data center and the PoP node of the CP.

$$\sum_{v \in N_{ai}^{V}} q_{rv} X_{vd} \leq Q_{rad}, \qquad \forall\ r \in R,\ a \in A,\ d \in D_i \tag{A.15}$$

$$Q_{rad} = Q_{rad} - \sum_{v \in N_{ai}^{V}} q_{rv} X_{vd}, \qquad \begin{array}{l} \forall\ r \in R,\ a \in A, \\ d \in D_i \end{array} \tag{A.16}$$

Constraint (A.15) ensures that the total amount of a resource required by all VMs assigned to a data center must not exceed its residual capacity. Constraint (A.16) updates this residual capacity after each VNR is successfully mapped.

$$B_{\overrightarrow{te}} = B_{\overleftarrow{te}}, \qquad \forall\ e \in L_{ti}^{S} \tag{A.17}$$

Constraint (A.17) states that the bandwidth capacity of a channel of a link type is the same in both directions on every substrate link.

$$\sum_{m \in N_i^S} f_{nm}^{uv} + X_{vn} b_l = \sum_{m \in N_i^S} f_{mn}^{uv} + X_{un} b_l, \quad \forall u, v \in N_i^V,\ v \neq u, n \in N_i^S,\ l = uv \in L_i^V \tag{A.18}$$

Constraint (A.18) guaranties the flow conservation for every amount of traffic $l = uv$ from VM u to VM v, with a bandwidth demand b_l routed on substrate link $e = mn \in L_i^S$.

$$\sum_{\varphi \in \mathcal{K}_e} \sum_{l \in L_{ti}^V} b_l Y_{l\varphi} + \sum_{\gamma \in \mathcal{K}_e} \sum_{k \in L_{t\phi_i}^V} b_k Z_{k\gamma} \leq B_{te}, \quad \forall\ e \in L_i^S,\ t \in T \tag{A.19}$$

$$B_e = B_e - \left(\sum_{\varphi \in \mathcal{K}_e} \sum_{l \in L_{it}^V} b_l Y_{l\varphi} + \sum_{\gamma \in \mathcal{K}_e} \sum_{k \in L_{t\phi_i}^V} b_k Z_{k\gamma} \right), \quad \forall\ e \in L_{ti}^S,\ t \in T \tag{A.20}$$

Constraint (A.19) ensures that the total bandwidth demand of all virtual links routed on a substrate link must not exceed the residual bandwidth capacity of the corresponding channel. Constraint (A.20) updates this residual capacity after a VNR is successfully mapped.

$$\sum_{d \in D_i} \sum_{c \in D_i} \sum_{\varphi \in \mathcal{P}_{dc}} \sum_{e \in \varphi} d_e Y_{l\varphi} \leq \delta_l,\ \forall\ l \in L_{ti}^V,\ t \in T \tag{A.21}$$

$$\sum_{d \in D_i} \sum_{\gamma \in \mathcal{P}_{d\phi_i}} \sum_{e \in \gamma} d_e Z_{l\gamma} \leq \delta_k, \quad \forall\ k \in L_{t\phi_i}^V,\ t \in T \tag{A.22}$$

Constraints (A.21) and (A.22) ensure that the restriction on the maximum delay allowed for a virtual link is not violated.

$$X_{vn} \in \{0,1\} \qquad \forall\ v \in N_{ai}^V,\ a \in A,\ n \in N_i^S \tag{A.23}$$

$$Y_{l\varphi} \in \{0,1\} \quad \forall\ l \in L_{ti}^V,\ t \in T,\ \varphi \in \mathcal{P}_{dc},\ d, c \in D_i \tag{A.24}$$

$$Z_{k\gamma} \in \{0,1\} \quad \forall\ k \in L_{t\phi_i}^V,\ t \in T,\ \gamma \in \mathcal{P}_{d\phi_i},\ d \in D_i \tag{A.25}$$

$$\omega_d \in \mathbb{R}_{\geq 0} \qquad \forall\ d \in D_i \tag{A.26}$$

Constraints (A.23) - (A.26) express the domain of definition of each variable.

A.2 Cost parameters setup

Most of cost parameters setting follows the experimentation setup defined in (Larumbe et Sanso, 2013), but they are specified for an hour time period. We chose the delay as the first optimization priority of the multi-objective function, by using the same weight attribution as in (Larumbe et Sanso, 2013). The unit computational resource cost is randomly determined in the interval $[1, 5]/core/h for CPU resource, $([1, 5]/10)/GB/h for memory resource and $([1, 5]/100)/GB/h for disk resource. The unit bandwidth cost for each link type is randomly chosen in the interval $([5, 10]/1000)/Mbps/h, and the corresponding extra unit bandwidth cost is estimated at 25% of the initial unit cost. The delay penalty is set to $1.15/(ms/packet)/h. The average power consumed by a VM assigned to a data center is randomly set in the interval [200 300] W (with 60% of the energy for CPU resource and 40% for disk resource). The unit electricity cost is randomly distributed in the interval $[30 70]/MWh. The PUE of a data center is 1.5. The amount of CO_2 emissions in a data center is calculated by summing values in the set {10 66 443 960} g/KWh, each of them first multiplied by a greenness factor randomly set between 0 and 1. The penalty for emitting CO_2 in a data center is defined as $1000/tonne.

CHAPITRE 6 ARTICLE 3 : AN EFFICIENT APPROACH BASED ON ANT COLONY OPTIMIZATION AND TABU SEARCH FOR A RESOURCE EMBEDDING ACROSS MULTIPLE CLOUD PROVIDERS

Auteurs : Marieme Diallo, Alejandro Quintero et Samuel Pierre.
Revue : Soumis dans le journal *IEEE Transactions on Cloud Computing*, en Novembre 2018.

Abstract

In cloud computing, a fundamental management problem with the Infrastructure as a Service (IaaS) model lies in the efficient embedding of computational and networking resources onto distributed virtualized infrastructures owned by independent cloud providers (CPs). In such a context, this issue usually referred to as the Virtual Network Embedding (VNE) problem, adds more complexity since the entire embedding process requires two mayor phases of operation: the multicloud virtual network requests (VNRs) splitting, followed by the intracloud VNR segments mapping. This paper focuses on the splitting phase problem, by proposing a VNRs splitting strategy formalized as an Integer Linear Program (ILP) model, with the objective of improving the performance and QoS of resulting mapped VNR segments, while minimizing the resource provisioning expenditures. As the VNE is classified as an NP-hard problem, a hybrid metaheuristic approach based on the Ant Colony Optimization (ACO) combined with the Tabu Search (TS) as local search operator, is proposed in order to find good feasible solutions in reasonable time. The simulation results show the efficiency of the proposed approach, which generates, in a highly reduced computing time, solution costs very close to the exact solution, with an average cost gap ranging from 0% to a maximum of 3.42%.

6.1 Introduction

Over the last few years, cloud computing has gained a significant popularity as a cost-effective utility computing model (Armbrust *et al.*, 2010; Zhang *et al.*, 2016a), by allowing small businesses to rent distributed configurable resources (computational and networking) offered as a large-scale service model, without huge investments in the operation and the maintenance. Among all the cloud service models, the Infrastructure as a Service (IaaS) has become nowadays the most widely adopted (Manvi et Shyam, 2014). In this model, a Service

Provider (SP) can lease, from one or more Cloud Providers (CPs), virtualized infrastructure layer resources (processing, storage, network access, routing services, etc.), packaged into interconnected virtual machines (VMs) and assembled as a virtual network request (VNR), in order to build heterogeneous virtual networks that will offer customized service applications to its end users.

Despite its successful adoption, the IaaS model faces a fundamental resource management challenge lying in the efficient and dynamic embedding of VNRs, where heterogenous resources are allocated to host VMs in specific substrate data centers (DCs), and to route virtual links (VLs), representing the exchanged traffic between VMs, onto substrate paths between the hosting DCs. This problem, usually referred to as the well-known NP-hard Virtual Network Embedding (VNE) problem (Fischer *et al.*, 2013; Zhang *et al.*, 2016a), has been only recently addressed in the literature (Houidi *et al.*, 2011; Chaisiri *et al.*, 2012; Samuel *et al.*, 2013; Ran *et al.*, 2016; Leivadeas *et al.*, 2013; Li *et al.*, 2016; Aral et Ovatman, 2016; Mechtri *et al.*, 2017; Dietrich *et al.*, 2015; Gong *et al.*, 2016; Mano *et al.*, 2016), in the case of a multi-cloud service where the substrate cloud infrastructures are owned by multiple independent CPs (Grozev et Buyya, 2012; Rafael *et al.*, 2012). Such a context adds more complexity and scalability issues, since the entire VNE process requires a hierarchical approach where two major phases of operation are performed, each of them dealing with an NP-Hard problem, with different purposes and resolution techniques: the multicloud VNRs splitting phase, followed by the intracloud VNR segments mapping phase. In the first phase, the SP generally mandates a virtual network provider (VNP), acting as a virtual brokerage service on behalf of the SP (Fischer *et al.*, 2013; Rabah *et al.*, 2015), which uses a splitting strategy to optimally select eligible CPs based on the SP's goals and requirements, and partitions the VNRs into segments. In the second phase, each selected CP uses a mapping approach to embed the assigned VNR segments into its intracloud network (Chowdhury *et al.*, 2012; Larumbe et Sanso, 2013; Melo *et al.*, 2015; Ayoubi *et al.*, 2016; Amokrane *et al.*, 2015; Justafort *et al.*, 2016; Hesselbach *et al.*, 2016; Khan *et al.*, 2016; Cao *et al.*, 2017; Pyoung et Baek, 2018), in order to satisfy economic benefits, resource utilization-efficiency, energy-efficiency, survivability or QoS aspects.

In this paper, we focus in particular on the splitting phase problem, which is more challenging. Indeed, due to the non-interoperability between CPs, as well as the CP policies that restrict the VNP's access on detailed information on the substrate multicloud environment, it becomes difficult to generate optimal embedding configurations which must best satisfy the SP's requirements. Moreover, the splitting phase is similar to the multiway separator problem (Sanchis, 1989; Tao *et al.*, 1992) known as NP-complete. Therefore, exact methods, using for example common optimization solvers like (CPLEX, 2018), are not suitable to solve

such a problem since they can only scale up to small instances of the problem due to the exponential increase of the resolution time. Solving the splitting problem then requires the development of fast large-scale approximate methods based on (meta) heuristic algorithms, in order to obtain optimal or near-optimal solutions in a polynomial computing time.

However, existing approaches for the multicloud VNRs splitting problem generally propose an exact solution (Dietrich *et al.*, 2015), or have assumed that CPs would share some private information with the VNP (Leivadeas *et al.*, 2013; Samuel *et al.*, 2013). Most others either aim at minimizing the resource provisioning price for the SP during the splitting phase (Houidi *et al.*, 2011; Chaisiri *et al.*, 2012; Dietrich *et al.*, 2015; Ran *et al.*, 2016; Mano *et al.*, 2016) or only propose performance-based models (Mechtri *et al.*, 2017; Leivadeas *et al.*, 2013; Li *et al.*, 2016; Aral et Ovatman, 2016), which may not give opportunities to the SP to select CPs based on desired performance and QoS, while at the same time minimizing the expenditures.

In order to address the challenges mentioned above in embedding VNRs across multiple IaaS providers, we propose in this paper a VNRs splitting strategy based on a hybrid metaheuristic approach. The main goal is to efficiently split the VNRs and to improve the performance and QoS of the resulting VNR segments that are mapped onto the selected cloud infrastructures, while minimizing the resource provisioning expenditures for the SP. The key contributions of this work are as follows:

— Our optimization strategy considers jointly VMs and inter-VMs traffic, including the required computational and networking resources, as well as the desired performance-based QoS constraints of the SP;

— The VNP's access level on information advertised in the multicloud environment, including the CPs' Point-of-Presence (PoP)(Dietrich *et al.*, 2015; Li *et al.*, 2016), is investigated according to CP policies. Such information are used to define the resource provisioning costs and constraints of the problem based on the price and availability of supplied resources in a high level of abstraction, as well as some performance guarantees (e.g. delay bounds (Larumbe et Sanso, 2013; Gong *et al.*, 2016)) disclosed to the VNP.

— We use an Integer Linear Program (ILP) to formalize the proposed splitting strategy as a minimization problem with constraints. The model is first solved with the exact method, named *Exact_Split*, performed with small instances of the problem by using the optimization solver (CPLEX, 2018);

— In order to solve large instances of the problem in reasonable time complexity, we propose a hybrid metaheuristic approach, named *ACO_TS_Split*, which is based on the evolutionary Ant Colony Optimization (ACO) approach (Dorigo et Stützle, 2010) combined with the Tabu Search (TS) algorithm (Glover, 1989, 1990) used as local

search (LS) operator. Such a hybridization approach allows us to take advantage on ACO's effective adaptive capabilities in optimization domain and on TS's efficiency in achieving (near) optimal solutions for various optimization problems via memory mechanisms.

The rest of the paper is organized as follows: Section 6.2 discusses relevant related work. Section 6.3 describes the proposed multicloud VNE model, as well as the related mathematical formulation. Section 6.4 presents the proposed *ACO_TS_Split* algorithm. Performance evaluation and simulation results are presented in Section 6.5. Finally, conclusion and future works are highlighted in Section 6.6.

6.2 Related work

The VNE problem, known as NP-hard, represents the main resource allocation challenge in IaaS-based cloud services (Manvi et Shyam, 2014; Zhang *et al.*, 2016a; Fischer *et al.*, 2013), especially in the case of a multicloud service (Grozev et Buyya, 2012; Rafael *et al.*, 2012). As a result, existing studies mostly propose (meta) heuristic resolution approaches. Here we discuss some related work on VNE over a multicloud network, as well as the principle of metaheuristics hybridization.

6.2.1 VNE over a multicloud network

VNE over a multi-domain has only recently been studied in the literature. It has been mainly addressed by authors in (Houidi *et al.*, 2011; Chaisiri *et al.*, 2012; Samuel *et al.*, 2013; Ran *et al.*, 2016; Leivadeas *et al.*, 2013; Li *et al.*, 2016; Aral et Ovatman, 2016; Mechtri *et al.*, 2017; Dietrich *et al.*, 2015; Mano *et al.*, 2016). Research works essentially focus on the VNRs splitting phase, which is more challenging. Indeed, existing solutions related to the single-domain VNE problem, largely addressed in the literature (e.g.(Chowdhury *et al.*, 2012; Larumbe et Sanso, 2013; Melo *et al.*, 2015; Ayoubi *et al.*, 2016; Amokrane *et al.*, 2015; Justafort *et al.*, 2016; Hesselbach *et al.*, 2016; Khan *et al.*, 2016; Cao *et al.*, 2017; Pyoung et Baek, 2018)), are usually adopted for the mapping phase.

Most VNRs splitting strategies aim at minimizing the expenditures for the SP (Houidi *et al.*, 2011; Chaisiri *et al.*, 2012; Dietrich *et al.*, 2015; Gong *et al.*, 2016; Ran *et al.*, 2016; Mano *et al.*, 2016). Chaisiri *et al.*(Chaisiri *et al.*, 2012) used a multistage stochastic integer programming model solved with different resolution approaches based on deterministic equivalent formulation, sample-average approximation and Benders decomposition, in order to propose a resource provisioning strategy that minimizes the total price for the SP. Works in (Dietrich

et al., 2015; Gong *et al.*, 2016; Mano *et al.*, 2016) studied the limited access level to information of the VNP before proceeding to the splitting phase. Dietrich *et al.* (Dietrich *et al.*, 2015) for example have investigated the suboptimality of multi-domain VNE with limited information disclosure and used the location of CPs' peering nodes and peering links to propose an exact ILP splitting model which minimizes the total resource provisioning price for the SP. Some works also considered the CPs' revenue maximization, like in (Samuel *et al.*, 2013), where the authors introduce a distributed protocol for VNE problem in multiple substrate networks coordinating the participating CPs through competitive pricing mechanisms. However, most of these proposals are only focused on VNRs splitting strategies at the lowest price for the SP, without optimizing performance and QoS of embedded VNR segments.

On the other hand, the proposals in (Leivadeas *et al.*, 2013; Li *et al.*, 2016; Aral et Ovatman, 2016; Mechtri *et al.*, 2017) are among the few performance-based approaches for VNRs splitting. Leivadeas *et al.* (Leivadeas *et al.*, 2013) proposed a splitting strategy based on Iterated Local Search (ILS) algorithm, where resource provisioning costs are defined based on the scarcity and the average utilization, which are generally concealed by the CPs (Dietrich *et al.*, 2015; Mano *et al.*, 2016) and, more importantly, may not necessarily reflect the performance of the CP's substrate network. Mechtri *et al.* (Mechtri *et al.*, 2017) have recently proposed a generic resources embedding model in distributed and hybrid cloud environments, solved with a heuristic algorithm based on graph decomposition and bipartite graph matching.

Based on the literature, most multicloud VNRs splitting strategies, even though they can propose sophisticated exact or heuristic splitting schemes, either tend to only minimize the resource provisioning price for the SP, or only propose performance-based approaches. Our proposal on the other hand tries to achieve both purposes, by giving opportunities to the SP to select CPs based on desired performance and QoS, while at the same time minimizing the resource provisioning expenditures.

6.2.2 Hybrid metaheuristics

Metaheuristic approaches are widely adopted for many hard optimization problems in order to find the best possible solutions in a reasonable time. In the recent years, one technique behind the principle of "hybridization" (Blum *et al.*, 2011) has gained a lot of attention in research communities, which generally consists in combining two different optimization methods in order to exploit the complementary potential of each of them. Hybridization techniques can use an integrative combination, where one of the methods is an integral part of the other, or a collaborative combination, where the algorithms are executed separately (sequentially, intertwined or in parallel), but one uses the results of the other. Some works

have proved the benefits of using hybrid metaheuristics to solve the VNE problem (Justafort *et al.*, 2016; Inführ et Raidl, 2016; Wang *et al.*, 2017a; Araújo *et al.*, 2018), mostly combining greedy or population based algorithms with a LS improvement operator, with different VNE objectives. Infuhr *et al.* (Inführ et Raidl, 2016) proposed a memetic approach combining a genetic algorithm (GA) with the variable neighborhood search (VNS) descent algorithm, used as local improvement technique for intensification. Authors in (Wang *et al.*, 2017a) used a hybrid Particle Swarm Optimization (PSO) VNE approach, which combines PSO, TS and simulated annealing algorithms, in order to improve the acceptance ratio and the CP's revenues. Araujo *et al.*(Araújo *et al.*, 2018) proposed an online VNE strategy based on Greedy Randomized Adaptative Search Procedure (GRASP) algorithm combined with Reduced VNS metaheuristics, for guaranteeing minimal delays, energy efficiency and maximal balancing in a multi-domain environment. Authors in (Justafort *et al.*, 2016) used a hybrid approach based on ILS and TS algorithms for solving the VMs placement problem in an intercloud environment, in order to minimize the carbon footprint.

Among the evolutionary metaheuristics, ACO algorithms have proven to be effective in addressing the VNE problem, due to their adaptation capabilities (Chang *et al.*, 2013; Guan *et al.*, 2015; Zhu et Wang, 2016; Wang *et al.*, 2017b; Zong *et al.*, 2018). However, solutions with the basic ACO can often be trapped in local optima (Dorigo et Stützle, 2010; Wang *et al.*, 2017b). Therefore, different techniques, such as LS processes, can be hybridized with ACO in order to maintain a high quality of solutions by better exploring the search space.

6.3 Multicloud VNE problem statement

This section briefly describes the multicloud VNE problem and the information repository used in our framework, before presenting the modeling and the mathematical formulation related to the splitting phase problem. Afterward, the adopted intracloud mapping approach performed by each CP for the mapping phase is briefly described, with more details available in Appendix B.

6.3.1 Problem overview

The global multicloud VNE process is described in Fig. 6.1. The problem consists in efficiently and dynamically embedding VNRs onto multiple cloud networks owned by independent CPs.

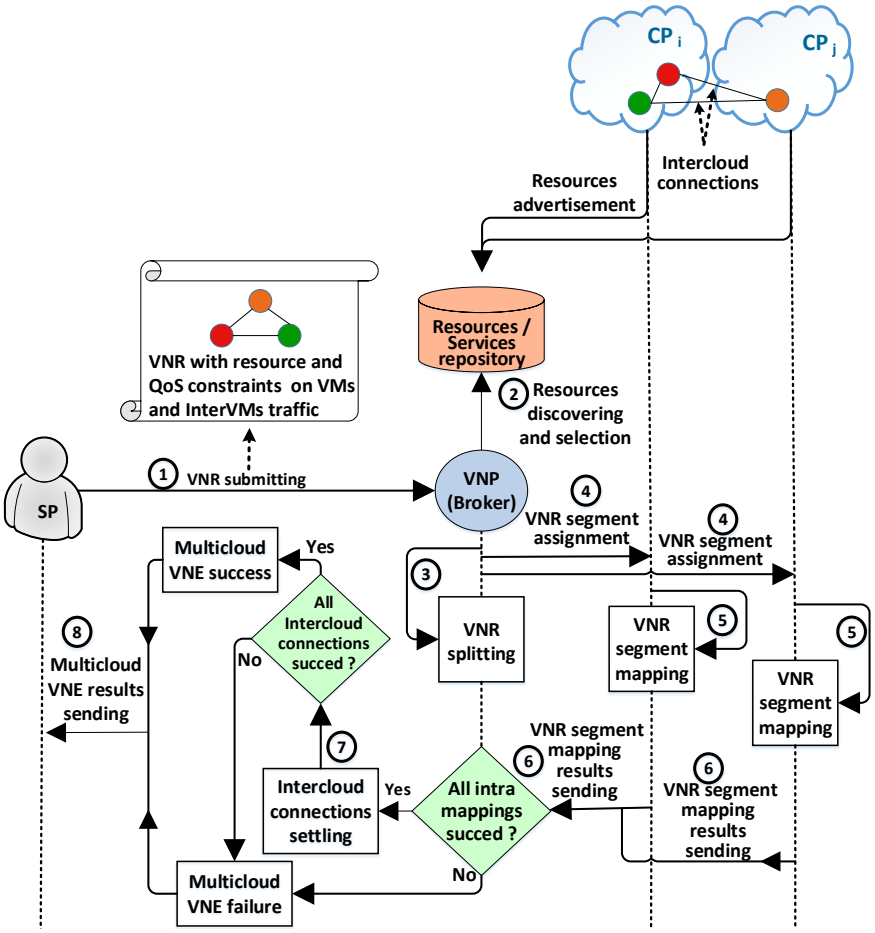

Figure 6.1 Multicloud VNE process

VNRs are submitted in a high level of abstraction by the SP, as a set of VMs interconnected by VLs representing the exchanged traffic between VMs (Fischer *et al.*, 2013; Zhang *et al.*, 2016a). VMs and VLs are specified with constraints related to required resources and QoS criteria (e.g., processing, memory, storage, bandwidth, QoS parameters, geographic footprint, etc.), that the embedding process has to satisfy. The objective for the SP is to efficiently lease heterogenous resources in order to host VMs in specific DCs, and to route VLs onto adequate substrate paths, so that the desired performance and QoS are achieved at the lowest possible resource provisioning expenditure.

The SP will generally rely on a virtual brokerage service, the VNP (Rabah *et al.*, 2015), which discovers and selects from a repository framework a set of advertised resources and services, assembled from multiple CPs at different provisioning costs. In most cases, to fulfill all the SP's requirements, the VNP will need to optimally split the VNRs among eligible CPs, by evaluating the most cost-effective resource provisioning depending on their availability and

price, while considering QoS restrictions specified in the request. Thereafter, each selected CP receiving a particular VNR segment maps it onto its infrastructure by using an appropriate optimized intracloud mapping method. In the case all VNR segments mappings succeed, the VNP will subsequently interconnect all segments via appropriate intercloud links (Leivadeas *et al.*, 2013) and send the overall multicloud VNE results to the SP.

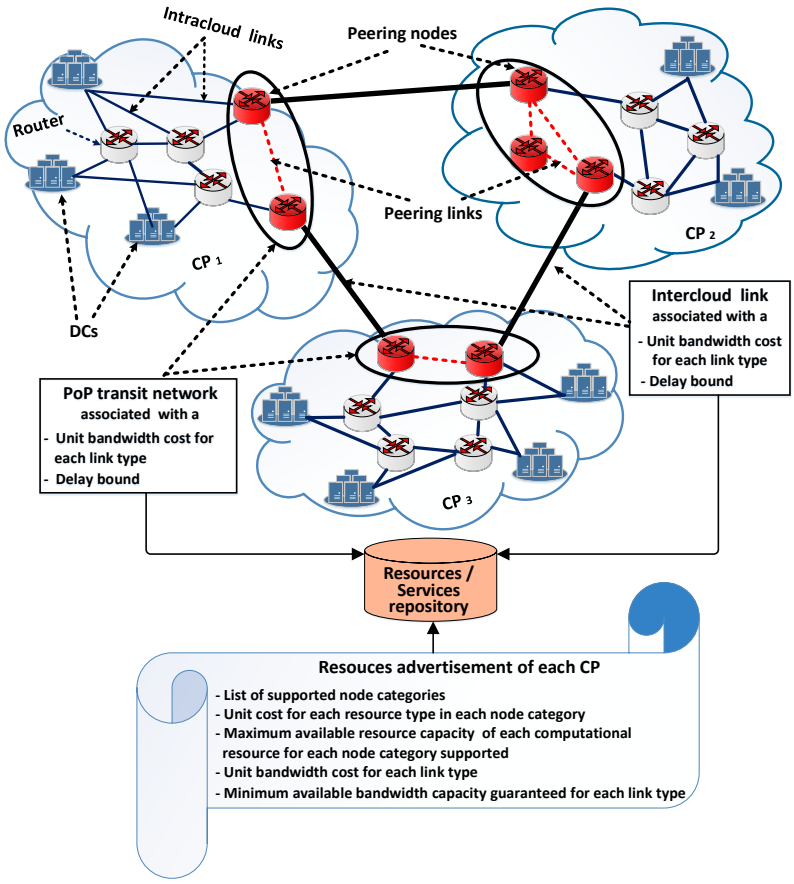

Figure 6.2 Multicloud environment and Information repository framework

6.3.2 Information repository framework and assumptions

In order to avoid inefficient embedding solutions, the information repository framework needs to be investigated (Amazon, 2017; Larumbe et Sanso, 2013; Dietrich *et al.*, 2015; Doverspike *et al.*, 2010; Chaisiri *et al.*, 2012; Zhang *et al.*, 2016a; Li *et al.*, 2016), in order to enrich the best possible the VNP's visibility on the multicloud environment, while complying with the CP policies.

Fig. 6.2 illustrates the results of this investigation we use in our splitting strategy. Note that as previously mentioned, detailed information on intracloud networks, such as network

topologies, router-level connectivities, number of DCs owned by CPs and their interconnectivity, number of instances of available resources and their utilization, as well as detailed transit network topologies, are not disclosed to the VNP (Dietrich *et al.*, 2015; Mano *et al.*, 2016; Gong *et al.*, 2016). Nevertheless, some information about transit networks (PoPs) of CPs can be accessible in a high level of abstraction (Dietrich *et al.*, 2015; Doverspike *et al.*, 2010). Those information can include an oversimplified view of PoP networks associated with an accumulated transit cost, as well as some traffic statistics and performance guarantees such as the estimated delay bound on an intercloud link and for transiting through a PoP network (Larumbe et Sanso, 2013; Gong *et al.*, 2016).

Moreover, a CP typically classifies its substrate nodes (servers located in DCs) into different categories (e.g. (Amazon, 2017)), each of them having common set of functional attributes (node type, operating system, virtualization environment, geographic footprint, QoS parameters, etc) and being associated with a set of computational resource types (CPU, disk space, memory, etc.) (Houidi *et al.*, 2011; Leivadeas *et al.*, 2013; Zhang *et al.*, 2016a). Similarly, substrate links are also classified into different types (e.g. VLAN, L3/L2 VPN, etc.). Computational and networking (i.e. bandwidth) resources are associated with an unit monetary cost and a certain capacity that the CP keeps dynamically updated. A CP can supply the maximum or the minimum of those resource capacities it can offer or guarantee (Chaisiri *et al.*, 2012), without revealing details about the distribution and utilization inside its intra-cloud network. To simplify the matching step between offered and demanded resources, we assume that VMs and VLs are specified with characteristics at the same level as substrate nodes and substrate links respectively. Each VM is classified into a specific category of node, which includes the desired location, and is defined with a set of computational resource types (i.e. CPU, disk space, memory), each of them associated with the corresponding amount of demand. Each VL is also associated with a link type, as well as an amount of bandwidth demand and a maximum delay allowed on the link. The latter is imposed by the SP for delay-sensitive applications.

For sake of simplicity, we only consider one PoP network per CP. Also, we did not consider in this work the path splitting scenario, as the VNP does not know the residual bandwidth capacity of intercloud paths.

Table 6.1 Notation for the VNRs splitting problem

Symbols	Description
	Global sets
A	Set of node categories
T	Set of link types
R	Set of computational resource types
	Multicloud substrate network
G^S	Graph representing the multicloud substrate network
I	Set of CPs
Θ	Set of PoP transit networks
N^S	Set of substrate nodes in G^S $(N^S = I \bigcup \Theta)$
L^S	Set of substrate links in G^S
Q_{rai}	Maximum available capacity of resource $r \in R$ that CP $i \in I$ can offer for nodes of category $a \in A, (Q_{rai} \in \mathbb{N})$
B_{ti}	Minimum available bandwidth capacity that CP $i \in I$ can guarantee for links of type $t \in T$ $(B_{ti} \in \mathbb{N})$
d_e	Average delay bound of substrate link $e \in L^S$
d_ϕ	Average delay bound for transiting through PoP network $\phi \in \Theta$
d_φ	Average delay bound of path $\varphi \in \mathcal{P}_{ij},\ i,j \in I$
$\mathcal{P}_{ij}$	Set of all paths between CP i and CP j
$\mathcal{O}_\varphi$	Set of all PoP transit networks in Θ spanned by path $\varphi \in \mathcal{P}_{ij},\ i,j \in I$ $(\ \forall \varphi \in \mathcal{P}_{ii},\ i \in I,\ \mathcal{O}_\varphi = \emptyset)$
	Virtual network request
G^V	Graph representing the VNR
N^V	Set of VMs in G^V
L^V	Set of VLs in G^V
N_a^V	Set of VMs of category of node $a \in A$ $(\bigcup_{a \in A} N_a^V = N^V)$.
L_t^V	Set of VLs of type $t \in T$
q_{rv}	Amount of resource $r \in R$ required by VM $v \in N^V$ $(q_{rv} \in \mathbb{N}_1)$
w_{rv}	Weight of resource $r \in R$ required by VM $v \in N^V$
b_l	Bandwidth demand of VL $l \in L^V$
δ_l	Maximum delay allowed for VL $l \in L^V$
Γ^V	Lifetime of a VNR

Table 6.1 Notation for the VNRs splitting problem

	Costs
$\mathcal{C}^N_{rai}$	Node provisioning quota of CP $i \in I$ for nodes of category $a \in A,\ r \in R$
$\mathcal{C}^L_{ti}$	Intracloud link provisioning quota of CP $i \in I$ for links of type $t \in T$
$\mathcal{C}^L_{t\varphi}$	Intercloud link provisioning quota of path $\varphi \in \mathcal{P}_{ij}$, $i, j \in I$, for links of type $t \in T$
$g^N_{rai}(\psi)$	Unit price of resource $r \in R$ supplied by CP $i \in I$ for nodes of category $a \in A$, during the time period ψ
$g^L_{ti}(\psi)$	Unit bandwidth price supplied by CP $i \in I$ for links of type $t \in T$, during the time period ψ
$g^L_{te}(\psi)$	Unit bandwidth price of substrate link $e \in L^S$ for links of type $t \in T$, during the time period ψ
$g^L_{t\phi}(\psi)$	Unit bandwidth price for transiting through PoP network $\phi \in \Theta$ for links of type $t \in T$, during the time period ψ
$g^L_{t\varphi}(\psi)$	Unit bandwidth price of path $\varphi \in \mathcal{P}_{ij},\ i, j \in I$ for links of type $t \in T$ during the time period ψ
	Decision variables
X_{vi}	Binary variable set to **1** if VM $v \in N^V_a$ is assigned to CP $i \in I$; **0** otherwise
$Y_{l\varphi}$	Binary variable set to **1** if VL $l \in L^V_t$ is assigned to path $\varphi \in \mathcal{P}_{ij},\ i, j \in I,\ t \in T$; **0** otherwise

6.3.3 VNRs splitting problem modeling and formulation

Here, the modelings of the multicloud substrate network and the VNR are presented, as well as the mathematical formulation related to the proposed splitting strategy.

6.3.3.1 Notation and modeling

Table 6.1 describes all the notation we used in our splitting model. The multicloud substrate network is modeled based on the given information repository framework, as a weighted undirected graph $G^S = (N^S, L^S)$, where N^S is the set of substrate nodes and L^S the set of substrate links (intercloud links). Peering nodes and peering links are aggregated into a single macro node, representing the PoP network of CPs in a high level of abstraction. Note that the PoP nodes are only used to access or transit through the network of a CP. The node and link provisioning quotas ($\mathcal{C}^N_{rai}$, $\mathcal{C}^L_{ti}$ and $\mathcal{C}^L_{t\varphi}$ parameters) represent the availability of computational and networking resources, which are weighted according to the maximum amount of these resources offered by the CPs. Each unit price of resources is given per time period, which in this work we assume fixed for each given period.

The VNR is also modeled as a weighted undirected graph $G^V = (N^V, L^V)$, where N^V and L^V are respectively the set of VMs and VLs. The maximum allowed delay δ_l of VL $l \in L^V$ adds constraints on the assignment of communicating VMs, in order to avoid traffic delay violations.

6.3.3.2 Multicloud resource provisioning costs definition

The following are the formulas that define each term and cost included in the objective function of the proposed ILP splitting model.

The node provisioning quota per CP is given by equation (6.1), as the maximum available capacity of each resource the CP can offer for each category of nodes, normalized into the range [1, 2] by dividing it by Q_{ra}^{max} (or by the value 1 to avoid division by zero):

$$C_{rai}^{N} = \frac{Q_{rai}}{\max\left(Q_{ra}^{max}, 1\right)} + 1 \ , \quad \forall\ r \in R,\ a \in A,\ i \in I \tag{6.1}$$

where

$$Q_{ra}^{max} = \max_{i \in I} \ \left(Q_{rai}\right) \ , \qquad \forall\ r \in R,\ a \in A \tag{6.2}$$

Equation (6.3) specifies the importance of each of the resources r requested by VM v, by associating each of them with a weight w_{rv} normalized into the range [1, 2]. Parameters q_r^{max} and q_r^{max} represent respectively the lowest and highest amount of resource r ever requested by a VNR, based on previous statistics collected by the VNP:

$$w_{rv} = \frac{q_{rv} - q_r^{min}}{q_r^{max} - q_r^{min}} + 1 \ , \qquad \begin{array}{l} \forall\ r \in R,\ v \in N^V, \\ q_r^{min} \leq q_{rv} \leq q_r^{max} \end{array} \tag{6.3}$$

The intracloud link provisioning quota per CP is given by equation (6.4), as the minimum available bandwidth capacity the CP can guarantee for each link type, normalized into the range [1, 2] by dividing it by B_t^{max} (or by the value 1 to avoid division by zero):

$$C_{ti}^{L} = \frac{B_{ti}}{\max(B_t^{max}, 1)} + 1 \ , \qquad \forall\ t \in T,\ i \in I \tag{6.4}$$

where

$$B_t^{max} = \max_{i \in I} \ \left(B_{ti}\right), \qquad \forall\ t \in T \tag{6.5}$$

The intercloud link provisioning quota of each path between a pair of CPs is given by equation (6.6), as the minimum intracloud link provisioning quota between the two endpoint CPs of the path:

$$C_{t\varphi}^{L} = \min\left(C_{ti}^{L}, C_{tj}^{L}\right), \quad \forall\ t \in T,\ i, j \in I, \varphi \in \mathcal{P}_{ij} \tag{6.6}$$

Note that in equation (6.6), i can be equal to j (in order to consider VLs whose corresponding connected VMs are assigned to the same CP i). In such cases, equation (6.6) will be equivalent to the intracloud link provisioning quota of CP i stated in equation (6.4).

Equation (6.7) calculates the total delay bound of a path:

$$d_{\varphi} = \sum_{e \in \varphi} d_e \; + \sum_{\phi \in O_{\varphi}} d_{\phi}, \qquad \forall \; \varphi \in \mathcal{P}_{ij}, \; i,j \in I \tag{6.7}$$

Equation (6.8) calculates the total unit bandwidth price of a substrate path. In the case the path connects two different CPs, the total unit bandwidth price accumulates the unit bandwidth price of the two corresponding CPs, plus those of each intercloud link spanned and PoP network transited.

$$g^L_{t\varphi}(\psi) = \begin{cases} g^L_{ti}(\psi) + g^L_{tj}(\psi) + \sum\limits_{e \in \varphi} g^L_{te}(\psi) + \sum\limits_{\phi \in O_{\varphi}} g^L_{t\phi}(\psi), \\ \qquad\qquad \forall \; \varphi \in \mathcal{P}_{ij}, \; i,j \in I, \; i \neq j, \; t \in T \\ \\ g^L_{ti}(\psi), \qquad \forall \; \varphi \in \mathcal{P}_{ii}, \; i \in I, \; t \in T \end{cases} \tag{6.8}$$

6.3.3.3 ILP formulation

The objective function of the proposed ILP splitting model is expressed as follows:

MIN

$$\begin{aligned} &\sum_{i \in I} \sum_{r \in R} \sum_{a \in A} \sum_{v \in N^V_a} \frac{\sum\limits_{\psi \in \Gamma^V} g^S_{rai}(\psi)\, q_{rv}}{w_{rv} C^N_{rai}} X_{vi} \quad + \\ &\sum_{i \in I} \sum_{j \in I} \sum_{\varphi \in \mathcal{P}_{ij}} \sum_{t \in T} \sum_{l \in L^V_t} \frac{\sum\limits_{\psi \in \Gamma^V} g^L_{t\varphi}(\psi)\, b_l}{C^L_{t\varphi}} Y_{l\varphi} \end{aligned} \tag{6.9}$$

where X_{vi} is a binary variable set to 1 if VM v is assigned to CP i, 0 otherwise, and $Y_{l\varphi}$ a binary variable set to 1 if VL l is assigned to path φ, 0 otherwise. The first term of the objective function represents the total node provisioning cost for assigning VMs to CPs. The second term represents the total link provisioning cost for assigning VLs to intracloud paths (i.e where i is equal to j) or intercloud paths (i.e. where i is different from j).

The model is subjected to the following constraints:

$$\sum_{i \in I} X_{vi} = 1, \qquad \forall\ v \in N_a^V,\ a \in A \tag{6.10}$$

Constraint (6.10) ensures that each VM must be assigned to exactly one CP.

$$\begin{aligned} Y_{l\varphi} \leq \frac{X_{ui} + X_{vj}}{2}, \quad & \forall\ l = uv \in L_t^V,\ t \in T, \\ & u, v \in N_a^V (v \neq u),\ a \in A, \\ & \varphi \in \mathcal{P}_{ij},\ i, j \in I \end{aligned} \tag{6.11}$$

Constraint (6.11) states the binary value of the variable $Y_{l\varphi}$ for each VL between two communicating VMs.

$$\sum_{i \in I} \sum_{j \in I} \sum_{\varphi \in \mathcal{P}_{ij}} Y_{l\varphi} = 1, \qquad \forall\ l \in L_t^V,\ t \in T \tag{6.12}$$

Constraint (6.12) ensures that each VL must be assigned to exactly one intracloud path, otherwise to an unique intercloud path.

$$\sum_{v \in N_a^V} q_{rv} X_{vi} \leq Q_{rai}, \qquad \forall\ r \in R,\ a \in A,\ i \in I \tag{6.13}$$

Constraint (6.13) ensures that the total amount of a resource required by all VMs assigned to a CP, must not exceed the maximum available capacity of the resource offered.

$$\sum_{i \in I} \sum_{j \in I} \sum_{\varphi \in \mathcal{P}_{ij}} d_\varphi Y_{l\varphi} \leq \delta_l, \quad \forall\ l \in L_t^V,\ t \in T \tag{6.14}$$

Constraint (6.14) ensures that the total delay on a path to which a VL is assigned must be less than the maximum delay allowed for the VL.

$$X_{vi} \in \{0, 1\} \qquad \forall\ v \in N_a^V,\ a \in A,\ i \in I \tag{6.15}$$

$$Y_{l\varphi} \in \{0, 1\} \qquad \forall\ l \in L_t^V,\ t \in T, \varphi \in \mathcal{P}_{ij},\ i, j \in I \tag{6.16}$$

Constraints (6.15) and (6.16) express the binary domain of variables.

The proposed ILP will guide the model to provide splitting solutions at the lowest price possible for the SP, while at the same time optimizing both the performance of VMs and inter-VM communication links by selecting CPs that can offer the highest (weighted) resource capacities (node and network) at the desired locations, and by avoiding the violation of delays specified by the SP. In addition, with the defined node provisioning cost, our approach considers all the resource types that can be requested by a VM, while weighting each of the corresponding amount of demand in order to efficiently allocate VMs according to the type

of application they support. For example, a VM more demanding in computation than in storage will be assigned to a provider offering much more CPU resource than disk resource (and vice-versa) at the lowest price possible, which can improve, from a global point of view, the performance of the VNRs and minimize the expenditures for the SP.

6.3.4 Intracloud VNR segments mapping phase

In our framework, we resolve the mapping phase problem by evaluating, for each selected CP i, the total revenue $\mathcal{R}_i$ and the total cost $\mathcal{C}_i$ of embedding, in order to maximize the profit P_i as follows:

MAX

$$P_i \;=\; \mathcal{R}_i \;-\; \mathcal{C}_i \qquad\qquad \forall\, i \in I \tag{6.17}$$

where

$$\begin{aligned}\mathcal{R}_i = \;& \sum_{r\in R}\sum_{a\in A}\sum_{v\in N_{ai}^V}\sum_{\psi\in\Gamma^V} g_{rai}^N(\psi)\, q_{rv} \;+ \\ & \sum_{t\in T}\sum_{l\in L_{ti}^V}\sum_{\psi\in\Gamma^V} g_{ti}^L(\psi)\, b_l \;+ \\ & \sum_{t\in T}\sum_{l\in L_{t\phi_i}^V}\sum_{\psi\in\Gamma^V} \left(g_{ti}^L(\psi) + g_{t\phi_i}^L(\psi)\right) b_l\end{aligned} \tag{6.18}$$

and

$$\mathcal{C}_i = \sum_{\psi\in\Gamma^V}\Big(\;\alpha\mathcal{C}_i^{\mathcal{S}} \;+\; \beta\mathcal{C}_i^{\mathcal{T}} \;+\; \lambda\mathcal{C}_i^{\mathcal{D}} \;+\; \varpi\mathcal{C}_i^{\omega} \;+\; o\mathcal{C}_i^{\mathcal{O}}\Big) \tag{6.19}$$

In equation (6.18), the parameters N_{ai}^V and L_{ti}^V respectively denote the set of VMs of node category $a \in A$ and the set of VLs of type $t \in T$ exclusively assigned to CP i. $L_{t\phi_i}^V$ represents the set of VLs of type $t \in T$ assigned to an intercloud link whose one of the endpoints is CP i. $g_{t\phi_i}^L(\psi)$ represents the unit extra bandwidth price for transiting through PoP network ϕ_i for links of type $t \in T$, during the time period ψ. The cost $\mathcal{C}_i$ in equation (6.19), detailed in Appendix B, is defined by adopting the multi-criteria solution stated in (Larumbe et Sanso, 2013), in which the total intracloud embedding cost (including respectively resource, traffic, delay, energy and environmental costs) is minimized for CP i.

Note that both SP's expenditure and CP's profit are depending on the lifetime Γ^V of VNRs. However, in this current work, we do not consider pricing policies such as "short-lived and high-cost" or "long-lived and low-cost".

6.4 The proposed *ACO_TS_Split* algorithm

In this section, we present the basic principles of ACO and TS algorithms, followed by the description of the proposed hybrid *ACO_TS_Split* approach, combining the two metaheuristics in order to solve the above ILP splitting model for large instances of the problem.

6.4.1 Basic principles of ACO and TS

Ant Colony Optimization (ACO) is a metaheuristic inspired by the behaviour of ant colonies (Dorigo et Stützle, 2010). A set of parallel artificial ants iteratively explore the search space until a predetermined stop criterion is reached. At each iteration, each ant incrementally constructs a solution by applying stochastic local decisions based on heuristic information and artificial pheromone trails. Once all ants have built their full solution, each ant (or the best one) deposits an amount of artificial pheromone on the components it (or the best one) has selected to build its solution. This allows ants to guide each other to the best regions in the solutions space, and to continuously refine the solutions to be obtained in future research. Besides ants' activity, the ACO algorithm includes two additional procedures: the *pheromone trail evaporation*, which helps the ants to discover new trajectories in the environment and not to be trapped in local optima, and the optional *daemon actions*, which can be implemented by centralized actions that can not be performed by single ants, such as adding more amount of pheromone on a promising track, or activating an improving local optimization search.

TS uses a LS method which explores the solution space by moving iteratively from a solution S to the best solution in the neighborhood of S. Contrary to classical descent methods, to overcome local optima and cycling problems, TS allows moves that deteriorate the actual best solution and stores temporarily forbidden moves in a tabu list. However, a tabu move can occasionally be allowed if it satisfies a specific *aspiration criterion*. The search stops whenever a given *stop criterion* is satisfied. The TS algorithm can also be enhanced by using adaptive memory mechanisms.

6.4.2 *ACO_TS_Split* algorithm

The proposed algorithm is presented in Fig. 6.3, which consists in an integrative combination of ACO and TS algorithms. *ACO_TS_Split* algorithm is executed for U^{max} iterations and uses a colony of K^{max} ants. At each iteration, each ant k constructs a solution S_k, by incrementally evaluating, for each possible solution component c to add to S_k, a probabilistic model $P_{S_k \oplus c}$ defined in equation (6.20) and using a heuristic information $\eta_{S_k \oplus c}$ and a pheromone value τ_c. After all ants have built their solution, TS is executed as a local

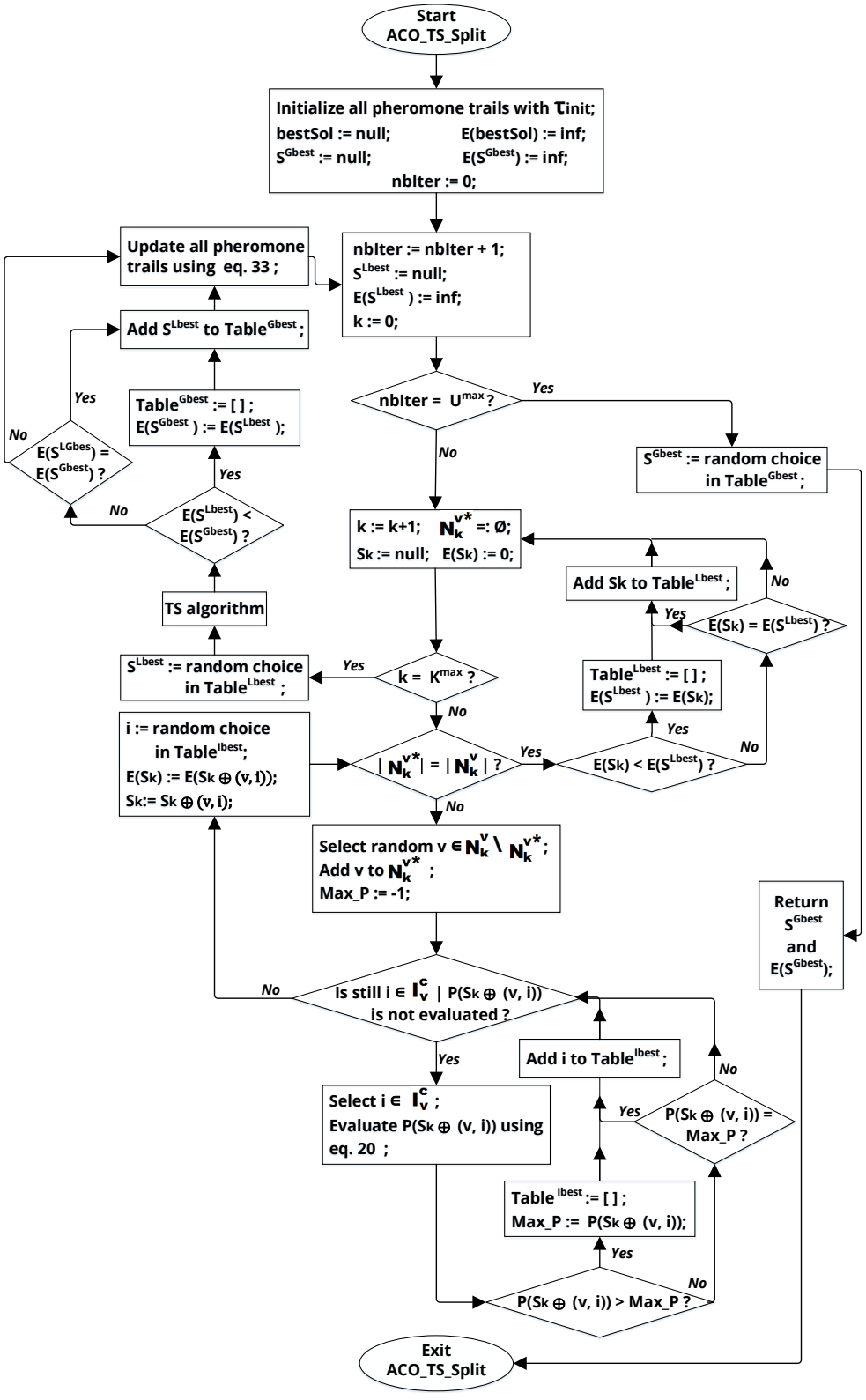

Figure 6.3 ACO_TS_Split algorithm

improvement operator on the best local solution S^{Lbest} of the iteration. The TS algorithm, introduced in our previous work (Diallo *et al.*, 2018c) and described in Appendix A, includes a LS process and an enhancing long-term memory (*Diversification*) mechanism. The LS is performed until S^{Lbest} remains unimproved during N^{max} iterations, then the *Diversification* is applied only once. Afterward, the best global solution S^{Gbest} is updated, as well as all pheromone trails by using equation (6.33). At the end *ACO_TS_Split* algorithm, S^{Gbest} is randomly chosen among all best solutions obtained so far.

Table 6.2 ACO parameters notation

U^{max}	Number of iterations for the ACO running
K^{max}	Number of ants in a colony
S_k	Solution constructed by ant k at each iteration
S^{Lbest}	Best local solution constructed at each iteration
S^{Gbest}	Best global solution so far constructed
$E(S_k)$	Evaluation cost of solution S_k
$P_{S_k \oplus (v,i)}$	Probabilistic model evaluated for selecting a solution component (v, i) to add to S_k
$\eta_{S_k \oplus (v,i)}$	Heuristic information used to evaluate $P_{S_k \oplus (v,i)}$
$\tau_{(v,i)}$	Pheromone trail on solution component (v, i)
α	Importance of the pheromone trail
β	Importance of the heuristic information
τ_{init}	Initial pheromone trail on all solution components
ρ	Evaporation rate of the pheromone trail
$N_k^{V^*}$	Set of VMs in N^V already assigned in solution S_k
$N_{ka}^{V^*}$	Set of VMs in N_a^V already assigned in solution S_k
I_v^C	Set of potential candidates CPs for assigning VM v
W_{kv}^*	Set of VMs in $N_k^{V^*}$ connected to VM v

Table 6.2 summarizes the parameters used for the proposed algorithm, with U^{max}, K^{max} and ρ, as well as N^{max} defined in Section 6.5.2 as results of parameterization tests. The solution construction, heuristic information and pheromone trails update of the ACO algorithm are detailed in the following.

6.4.2.1 ACO solution construction

A solution S of *ACO_TS_Split* is the assignment of each VM v of a VNR to a unique CP i, and the assignment of each VL l to a unique intracloud or intercloud path φ. Thus, a solution is determined by the settling of the binary variables X_{vi} and $Y_{l\varphi}$ that satisfies the constraints

of the proposed ILP splitting model. Thereby, in order to avoid unfeasible solutions, the cost of a solution S is defined by its evaluation cost $E(S)$, which includes the intrinsic cost related to the objective function in (6.9), plus the penalty costs particularly related to the possible violation of constraints (6.13) and (6.14).

Each ant k begins with an empty solution S_k, and incrementally adds a solution component to S_k until the whole solution is constructed. In our algorithm, we represent a solution component c as a pair (v, i), which assigns VM $v \in N_a^V$ to CP $i \in I_v^C$, as well as all VLs connecting VM v and VM $u \in W_{kv}^*$ to the suitable substrate paths. I_v^C denotes the set of potential candidate CPs for assigning VM $v \in N_a^V$ (i.e, the set of CPs in I that offer category of node $a \in A$). A solution component (v, i) added to S_k is represented by $S_k \oplus (v, i)$.

In order to select a new solution component (v, i) to add to S_k from all the possibilities, a VM v not yet assigned in S_k is randomly chosen and the probabilistic model $P_{S_k \oplus (v,i)}$ is evaluated for all potential candidates $i \in I_v^C$. Therefore, the ant k selects the solution component (v, i) that generates the highest value of $P_{S_k \oplus (v,i)}$, given by:

$$P_{S_k \oplus (v,i)} = \frac{\tau_{(v,i)}^{\alpha} \ \eta_{S_k \oplus (v,i)}^{\beta}}{\sum\limits_{j \in I_v^C} \tau_{(v,j)}^{\alpha} \ \eta_{S_k \oplus (v,j)}^{\beta}}, \tag{6.20}$$

where α and β are positive values representing respectively the relative importance of the pheromone trail and the heuristic information.

6.4.2.2 Heuristic information definition

The heuristic information is defined as follows:

$$\eta_{S_k \oplus (v,i)} = \frac{1}{E\left(S_k \oplus (v, i)\right)}, \quad \forall \ v \in N_{ka}^V / N_{ka}^{V^*}, \ i \in I_v^C \tag{6.21}$$

where $E(S_k \oplus (v, i))$ represents the new evaluation cost of solution S_k after adding component (v, i), defined as:

$$\begin{aligned} E\left(S_k \oplus (v, i)\right) \ = \ E\left(S_k\right) \ + \ \Delta_{E(S_k \oplus (v,i))}, \\ \forall \ v \in N_{ka}^V / N_{ka}^{V^*}, \ a \in A, \ i \in I_v^C \end{aligned} \tag{6.22}$$

Indeed, instead of entirely calculating at each time the new evaluation cost $E(S_k \oplus (v, i))$, only the evaluation cost difference $\Delta_{E(S_k \oplus (v,i))}$ generated by adding (v, i) to S_k is calculated and added to the old cost $E(S_k)$. This approach is more complicated to implement but it decreases significantly the computing time from complexity order $O(n^2)$ to $O(n)$, with n representing the total number of VMs in a request.

The evaluation cost difference $\Delta_{E(S_k \oplus (v,i))}$ is defined as follows:

$$\begin{aligned} \Delta_{E(S_k \oplus (v,i))} \;=\; & \Delta^{N}_{E(S_k \oplus (v,i))} + \; \Delta^{P^N}_{E(S_k \oplus (v,i))} \; + \\ & \Delta^{L}_{E(S_k \oplus (v,i))} \; + \; \Delta^{P^L}_{E(S_k \oplus (v,i))}, \\ & \forall\; v \in N^V_{ka}/N^{V^*}_{ka},\; a \in A,\; i \in I^C_v \end{aligned} \tag{6.23}$$

$\Delta^{N}_{E(S_k \oplus (v,i))}$ represents the intrinsic node cost difference from S_k to $S_k \oplus (v,i)$, given by:

$$\begin{aligned} \Delta^{N}_{E(S_k \oplus (v,i))} = \sum_{r \in R} \left(\frac{\sum\limits_{\psi \in \Gamma^V} g^{S}_{rai}(\psi)\, q_{rv}}{w_{rv} C^{N}_{rai}} \right), \\ \forall\; v \in N^V_{ka}/N^{V^*}_{ka},\; a \in A,\; i \in I^C_v \end{aligned} \tag{6.24}$$

$\Delta^{P^N}_{E(S_k \oplus (v,i))}$ represents the node penalty cost difference from S_k to $S_k \oplus (v,i)$, given by:

$$\begin{aligned} \Delta^{P^N}_{E(S_k \oplus (v,i))} = -10^9 \times \sum_{r \in R} \min(0, Q_{rai} - (Q^*_{rai} + q_{rv})), \\ \forall\; v \in N^V_{ka}/N^{V^*}_{ka},\; a \in A,\; i \in I^C_v \end{aligned} \tag{6.25}$$

where Q^*_{rai} represents the total amount of resource $r \in R$ of all VMs of category $a \in A$ assigned to CP i, i.e.:

$$Q^*_{rai} = \sum_{u \in N^V_a} q_{ru} X_{ui}, \qquad \forall\; i \in I \tag{6.26}$$

$\Delta^{L}_{E(S_k \oplus (v,i))}$ represents the intrinsic link cost difference from S_k to $S_k \oplus (v,i)$, given by:

$$\Delta^{L}_{E(S_k \oplus (v,i))} = \sum_{u \in W^*_{kv}} \left(\begin{array}{l} \dfrac{\sum\limits_{\psi \in \Gamma^V} g^{L}_{t_1 \varphi^1_{im}(l_1)}(\psi)\, b_{l_1}}{C^{L}_{t_1 \varphi^1_{im}(l_1)}} + \\ \\ \dfrac{\sum\limits_{\psi \in \Gamma^V} g^{L}_{t_2 \varphi^1_{mi}(l_2)}(\psi)\, b_{l_2}}{C^{L}_{t_2 \varphi^1_{mi}(l_2)}} \end{array} \right) \tag{6.27}$$

where m represents the CP i to which VM $u \in W^*_{kv}$ is assigned in S_k. l_1 and l_2 represent respectively VLs $l_1 = vu$ and $l_2 = uv$. t_1 and t_2 represent the types of link in T, respectively for l_1 and l_2.

$\varphi^1_{im}(l_1)$ (respectively $\varphi^1_{mi}(l_2)$) represents the path in $\mathcal{P}_{im}$ (respectively in $\mathcal{P}_{mi}$) to which the algorithm assigns VL l_1 (respectively l_2), by performing equations (6.28), (6.29) and (6.30), defined as follows, $\forall\; i, j \in I, l \in L^V_t$:

$$F_{ij}(l) \quad = \quad \{\varphi \in \mathcal{P}_{ij} \mid d_{\varphi} \leq \delta_l\} \tag{6.28}$$

$$F^1_{ij}(l) = \begin{cases} \left\{ \varphi^1 \in F_{ij}(l) \mid \sum\limits_{\psi \in \Gamma^V} g^L_{t\varphi^1}(\psi) = \min\limits_{\varphi \in F_{ij}(l)} \left(\sum\limits_{\psi \in \Gamma^V} g^L_{t\varphi}(\psi) \right) \right\} \text{ if } \; F_{ij}(l) \neq \emptyset \\ \\ \left\{ \varphi^1 \in \mathcal{P}_{ij} | \quad \sum\limits_{\psi \in \Gamma^V} g^L_{t\varphi^1}(\psi) = \min\limits_{\varphi \in \mathcal{P}_{ij}} \left(\sum\limits_{\psi \in \Gamma^V} g^L_{t\varphi}(\psi) \right) \right\} \; \text{ if } \; F_{ij}(l) = \emptyset \end{cases} \tag{6.29}$$

$$\varphi^1_{ij}(l) = \begin{cases} F^1_{ij}(l) & \text{if } \left| F^1_{ij}(l) \right| = 1 \\ \\ \varphi^1 \in F^1_{ij}(l) \mid \quad d_{\varphi^1} = \min\limits_{\varphi \in F^1_{ij}(l)} (d_{\varphi}) & \text{if } \left| F^1_{ij}(l) \right| > 1 \end{cases} \tag{6.30}$$

For each VL l of type t that has to be assigned to a path between two CPs i and j, equation (6.28) first selects all path $F_{ij}(l)$ in $\mathcal{P}_{ij}$ that do not violate the maximum delay on l. If such paths exist, equation (6.29) will choose those with the lowest accumulated unit bandwidth price, otherwise those in $\mathcal{P}_{ij}$ with the minimal price too, represented by $F^1_{ij}(l)$. If there is more than one satisfying path in $F^1_{ij}(l)$, equation (6.30) will select a path $\varphi^1_{ij}(l) \in F^1_{ij}(l)$ with the lowest delay.

$\Delta^{P^L}_{E(S_k \oplus (v,i))}$ represents the link penalty cost difference from S_k to $S_k \oplus (v,i)$, given by:

$$\Delta^{P^L}_{E(S_k \oplus (v,i))} = 10^9 \times \left| H^L_{S_k \oplus (v,i)} \right|, \; \forall \; v \in N^V_k / N^{V^*}_k, \; i \in I^C_v \tag{6.31}$$

where $H^L_{S_k \oplus (v,i)}$ represents the set of all VLs connected with v whose the maximum delay is violated in $S_k \oplus (v,i)$, given by:

$$\begin{aligned} H^L_{S_k \oplus (v,i)} = & \left\{ l = vu \in L^V \mid d_{\varphi^1_{im}(l)} > \delta_l \right\} \cup \left\{ l = uv \in L^V \mid d_{\varphi^1_{mi}}(l) > \delta_l \right\}, \\ & \forall \; u \in W^*_{kv}, \; \forall \; v \in N^V_k / N^{V^*}_k, \; i \in I^C_v \end{aligned} \tag{6.32}$$

6.4.2.3 Pheromone trail update

The pheromone trail is updated on all solution component (v, i) as follows:

$$\tau_{(v,i)} = (1-\rho)\tau_{(v,i)} + \Delta_{\tau_{(v,i)}}, \quad \forall\ v \in N^V,\ i \in I_v^C \tag{6.33}$$

where ρ is the evaporation rate (with $0 < \rho \leq 1$), and $\Delta_{\tau_{(v,i)}}$ represents the additional amount of pheromone deposited on component (v, i) belonging to the best local ant of the iteration, defined as:

$$\Delta_{\tau_{(v,i)}} = \begin{cases} \dfrac{1}{E(S^{Gbest})}, & \text{if } (v,i) \in S^{Lbest} \\ 0 & otherwise \end{cases} \tag{6.34}$$

The pheromone trail evaporation allows to avoid a too rapid convergence of the algorithm towards a sub-optimal region, and then to favor the exploration of new areas of the search space. In our parameterization tests, best results were obtained when pheromone updates are performed using the evaluation cost of the global best solution S^{Gbest}.

6.5 Numerical results

In this section, the efficiency of the proposed hybrid VNRs splitting strategy is evaluated through simulations. To this end, two main experiments are considered: *Experiment 1*, where the performance of *ACO_TS_Split* algorithm is evaluated in terms of quality of solutions obtained and computing time; and *Experiment 2*, where the performance of the proposed *ACO_TS_Split* splitting strategy is evaluated according to the acceptance rate and execution time, as well as some economic aspects, such as SP's expenditure and CP's profit.

ACO_TS_Split is implemented in C++ under Visual Studio 14, while *Exact_Split* is implemented in AMPL 64 bits with the CPLEX 12.6.3 solver. All simulations are run on a single server with an Intel Core i7 CPU at 4 GHz and 32 GB of RAM.

6.5.1 Experiments setup

Each of *Experiment 1* and *Experiment 2* is conducted considering 10 participating CPs in the multicloud:

Experiment 1, which compares the proposed hybrid approach *ACO_TS_Split* with *Exact_Split*, *ACO_Split* and *TS_Split*. *Exact_Split* is the optimal approach. *ACO_Split*

represents the proposed *ACO_TS_Split* approach without the hybridization with TS and *TS_Split* is the proposed TS algorithm starting with an initial random solution and detailed in Appendix B. Each of the *ACO_Split*, *TS_Split* and *ACO_TS_Split* algorithm is executed 10 times for each instance of the problem. We first consider in this experiment 14 instances of small sized VNRs, in which the number of interconnected VMs increases with an incremental 5-step, respectively from 5 to 70. Subsequently, we consider 18 instances of large sized VNRs, in which the number of VMs increases with an incremental 25-step, respectively from 75 to 500. For large instances, we compare only *ACO_TS_Split* and *TS_Split*. Indeed, the exact method could not perform large instances of the problem in a reasonable time. As for *ACO_Split*, without the enhancing TS as local improvement operator, the algorithm can easily be trapped in bad local optima, sometimes leading to non-feasible solutions due to some violated VLs delays.

Experiment 2, which compares different approaches named *Exact_Split_cost_perf*, *ACO_TS_Split_cost_perf*, *Exact_Split_cost*, *ACO_TS_Split_cost*, *Exact_Split_perf* and *ACO_TS_Split_perf*, according to the performance and economic metrics listed above. *Exact_Split_cost_perf* and *ACO_TS_Split_cost_perf* represent our splitting strategy respectively with the exact and approximate hybrid approaches, which consider both cost minimization and performance maximization. *Exact_Split_cost* and *ACO_TS_Split_cost* consider only the cost minimization (i.e. where the denominators in equation (6.9) have 1 as values and constraint (6.13) is removed). This could be the SP's choice or because the CPs have hidden all resource information except the prices. *Exact_Split_perf* and *ACO_TS_Split_perf* optimize only the performance maximization (i.e. where the numerators in equation (6.9) have 1 as values). Here, we run 50 simulations with the approaches performed with small sized VNRs, with 100 random incoming requests in each simulation. Each approach uses the same adopted method for the intracloud mapping phase detailed in Appendix B and implemented with the exact solution by using the CPLEX solver too.

All topologies and features of the substrate network and the VNRs are randomly generated using a MATLAB program and the parameters set in Table 6.3. Three types of computational resources are considered, which are CPU, memory and disk. VNRs topologies are randomly generated in a partial mesh, with 50% probability of interconnection between VMs. The multi-cloud substrate network interconnecting all participating CPs is generated based on ISPs topologies (Bhamare *et al.*, 2015; Doverspike *et al.*, 2010). CPs are interconnected in a half mesh connectivity. DCs in each substrate intra-cloud are interconnected through a backbone network, similar to the NSFNet topology (Amokrane *et al.*, 2013). Substrate nodes are located randomly at different geographic areas and each DC is connected to the

backbone network through the closest routers to its location. The number of substrate nodes in each CP is set to 25, with 20% probability of generating DC nodes and 80% of router nodes, the latter being only used for forwarding purposes. We consider that DCs are heterogeneous (can support different categories of node) with different resource capacities. Residual resource capacities for substrate nodes and substrate links, as well as resources advertised in the repository framework, are dynamically updated by CPs after a VNR has been mapped or an existing VNR has expired. The unit computational resource price is randomly determined in the interval $[5, 10]/core/h for CPU, $([5, 10]/10)/GB/h for memory and $([5, 10]/100)/GB/h for disk. The unit bandwidth price for each link type is randomly chosen in the interval $([25, 30]/1000)/Mbps/h for intra-cloud links and in $([30, 35]/1000)/Mbps/h for inter-cloud links and for transiting through PoP networks. Details related to the setup of cost parameters used to calculate the total embedding cost of CPs are available in Appendix B.

Table 6.3 Experiments parameters

	Value / distribution interval
Substrate network	
Number of substrate nodes per CP	25
Number of substrate links per CP	50 on average
Degree of nodes interconnectivity per CP	[3, 6]
DC CPU capacity per node category	[100, 500] cores
DC memory capacity per node category	[1000, 5000] GB
DC disk capacity per node category	[10000, 50000] GB
Intracloud link bandwidth capacity per link type	[4000, 5000] Mbps
Intracloud link delay bound	[0, 3] ms
Intercloud link / PoP transit delay bound	[0, 25] ms
Virtual network request	
VM CPU demand	[1, 40] cores
VM memory demand	[1, 400] GB
VM disk demand	[1, 4000] GB
VL bandwidth demand	[1, 20] Mbps
VL maximum delay allowed	[0, 300] ms
VNR average arrival rate	1 per 100 time units
VNR lifetime	[1000 100000] time units

6.5.2 Parameters of *ACO_TS_Split*

Preliminary tests were performed in order to define the optimal parameters of *ACO_TS_Split* algorithm, which mostly depend on the instances size: $K^{max} = 3 * \sqrt[3]{|N^V|} + I$, $\rho = 0.3$, $\alpha = 2$, $\beta = 1$. For small sized VNRs, $U^{max} = 10 * \sqrt[3]{|N^V|} + I$ and $N^{max} = 25 * \sqrt{|N^V| * |I|}$. For large sized VNRs, $U^{max} = 5 * \sqrt[3]{|N^V|} + I$ and $N^{max} = 25 * \sqrt{|N^V| * |I| / 4}$. U^{max} and N^{max} are reduced for large instances due to the negligible cost improvements in most cases, compared to the high resulting CPU time. Indeed, as VMs are interconnected in partial mesh, with n VMs the number of VLs (in both directions of communication) increases considerably in the range of $n(n-1)/2$. The space of feasible solutions is then significantly reduced because of the higher probability of violating some VLs delays, compared to the solution space with small sized VNRs. Therefore, it becomes useless for the heuristic to explore the space more than necessary.

6.5.3 Results analysis

The results obtained with *Experiment 1* and *Experiment 2* are presented in the following, with the size of requests expressed in number of interconnected VMs for all results.

Table 6.4 Comparison between *Exact_Split*, *ACO_Split*, *ACO_TS_Split* and *TS_Split* with small sized VNRs with *Experiment 1*

Exact_Split		*ACO_Split*		*ACO_TS_Split*				*TS_Split*	
				without Diversification		*with Diversification*			
Number of VMs	Mean CPU time (s)	Mean Cost_gap (%)	Mean CPU time (s)	Mean Cost_gap (%)	Mean CPU time (s)	Mean Cost_gap (%)	Mean CPU time (s)	Mean Cost_gap (%)	Mean CPU time (s)
5	4.3	0.12	0.53	0	0.8	0	0.81	0	0.9
10	84.4	0.45	1.22	0	1.54	0	1.65	0	1.13
15	442.2	1.02	2.46	0	2.81	0	2.92	0	2.8
20	918.1	2.52	3.65	0	4.76	0	4.99	0	5.71
25	2097.3	3.17	6.26	0	8.61	0	9.07	0	9.42
30	3025.4	3.78	8.83	1.84	14.17	0	15.03	0	22.2
35	4266.5	4.56	12.38	2.43	18.79	0	19.24	0	26.24
40	5849	5.53	17.33	1.81	26.5	0	27.73	0.21	37.32
45	10385.6	7.38	21.52	3.74	37.47	0.47	38.82	0.53	55.24
50	15328.5	9.8	26.05	3.62	45.72	2.35	46.64	2.29	69.53
55	19677.9	10.07	32.18	5.25	61.44	3.12	63.07	4.18	80.95
60	2210.8	12.51	37.86	6.41	72.76	2.82	74.47	3.04	88.46
65	33703.7	13.6	42.85	7.2	103.09	3.03	106.27	4.11	121.29
70	50120.1	20.43	49.1	7.18	128.06	3.42	130.68	3.85	156.14

Table 6.5 Comparison between *ACO_TS_Split* and *TS_Split* with large sized VNRs with *Experiment 1*

	ACO_TS_Split				*TS_Split*	
	without Diversification		*with Diversification*			
Number of VMs	Mean Cost_gap (%)	Mean CPU time (s)	Mean Cost_gap (%)	Mean CPU time (s)	Mean Cost_gap (%)	Mean CPU time (s)
75	5.24	133.01	1.16	135.25	1.06	159.76
100	5.92	159.21	1.16	161.84	1.2	173.5
125	6.63	188.5	1.23	192.42	1.2	193.7
150	6.54	217.32	2.27	221.32	2.53	215.13
175	7.16	229.37	2.45	233.53	3.05	246.31
200	7.67	242.65	1.37	246.86	1.6	269.59
225	7.88	269.85	2.5	274.44	2.53	255.62
250	8.1	321.25	3.8	327.23	3.92	295.78
275	10.39	378.12	2.62	384.24	3.04	424.43
300	9.11	433.86	3.16	440.11	3.02	496.48
325	9.38	516.91	3.82	523.24	3.55	659.97
350	10.42	621.23	3.88	627.62	4.6	766.14
375	10.54	813.28	3.56	819.69	5.3	821.84
400	11.53	970.36	4.93	977.89	4.8	984.83
425	11.9	1184.23	5.17	1192.19	5.19	1062.79
450	11.19	1441.18	4.32	1449.63	5.06	1464.1
475	12.57	1617.6	4.52	1627.14	5.9	1625.2
500	12.13	2098.87	4.03	2109.25	5.55	2036.13

6.5.3.1 *ACO_TS_Split* with Experiment 1

Table 6.4 and Table 6.5 show the results obtained with *Experiment 1*. Table 6.4 first reports, for each VNR instance, the mean execution time for *Exact_Split*. Then, for each of *ACO_Split*, *ACO_TS_Split* (without and with *Diversification*) and *TS_Split* approach, the mean solution cost gap from the exact method and the execution time are reported. *Cost_gap*, representing the average distance (in %) of the cost obtained with any of the heuristic solution from the exact solution, is calculated as:

$$Cost\_gap = 100 * \frac{(Heuristic - Exact)}{Exact}, \tag{6.35}$$

The proposed *ACO_TS_Split*, with the *Diversification* mechanism, is able to reach quasi-always the optimal solution, with zero cost gaps for VNRs with less than 45 VMs and a maximum of about 3.42%, giving then an average of about 1.09% considering all instances. *TS_Split* is as effective, generating an average cost gap of about 1.3%. *ACO_TS_Split* without *Diversification* gives higher cost gaps up to 7.2%, with a total average of about

2.82%. *ACO_Split* generates solutions with the least good qualities, with a total average of about 6.78% including gaps up to 20.43%.

As for the CPU time, it is generally correlated and increases on average according to the size of the instances. With the exact method, since CPLEX uses *Branch-and-Bound* methods, the CPU time increases exponentially as expected, from 4.3 s with 5 VMs to 50120.1 s with 70 VMs. The proposed heuristics meanwhile significantly reduce the CPU time, which becomes linear, besides being effective in solving such a combinatorial networking problem. Indeed, *ACO_TS_Split* without and with *Diversification* give respectively an average of CPU time of about 37.61 s and 38.67 s for all instances. Naturally, the CPU times are the lowest with *ACO_Split*, with an average of only 18.73 s. *TS_Split* gives the highest CPU times, with an average of about 48.38 s. Indeed, as described in Appendix A, the TS algorithm starting with a random (unfeasible) solution, needs much more iterations and *Diversification* restarts to find a good solution, which is more expensive in calculation time.

Tests reported in Table 6.5 with large sized VNRs confirm the previous results on the cost gap. Here, since we could not perform the exact method, the gap is calculated with the maximal solution cost ever found by the heuristic, for each instance. *ACO_TS_Split* with *Diversification* and *TS_Split* remain more efficient, with diversified *ACO_TS_Split* generating slightly better results, compared to the case without *Diversification* which results in gaps up to 12.57%. Regarding the CPU time, it is slightly the same for the three approaches, with *TS_Split* taking a little more time (total average of about 675.07 s) than *ACO_TS_Split* (total average of 663.55 s with *Diversification* applied).

Results with *Experiment 1* prove that the LS process hybridized with ACO significantly improves the solutions compared to the simple ACO algorithm, even if it is at the cost of more CPU time. This allows us to offer a very good tradeoff between quality of solutions and computing time.

6.5.3.2 *ACO_TS_Split* with Experiment 2

Results with *Experiment 2* are first given with the computing time complexity reported in Table 6.6. For each of the six approaches, the mean CPU time is given for each size of request. For the three exact approaches, we arbitrary set a time limit (TL) of splitting processing to 21,600 s and the percentage of request instances reaching the TL is also indicated. As expected, the processing time taken by the solver increases exponentially with the number of interconnected VMs. However, we can notice that *Exact_Split_cost* is the approach which converges the fastest towards the TL, which is reached by all requests with more than 45 VMs, without the proof of optimality of the obtained solution being established.

Exact_Split_perf and *Exact_Split_cost_perf* can provide optimal solutions with requests with up to respectively 65 VMs and 60 VMs, before the TL is 100% reached. Same observation can be noticed with the heuristic approaches, with *ACO_TS_Split_cost* taking much more computing times (with a total average of about 119.44 s), compared to *ACO_TS_Split_perf* and *ACO_TS_Split_cost_perf* generating respectively a total average of 19.67 s and 38.35 s. These results are due to the larger search space of solutions with *ACO_TS_Split_cost* (and *Exact_Split_cost*) because of less constraints considered in the model, which takes more time to compute all possible solutions.

Table 6.6 Computing time complexity of the splitting phase with *Experiment 2*

	Exact_ Split_ cost		*Exact_ Split_ perf*		*Exact_ Split_ cost_perf*		*ACO_ TS_ Split_ cost*	*ACO_ TS_ Split_ perf*	*ACO_ TS_ Split_ cost_perf*
Number of VMs	Mean CPU time (s)	% TL reached	Mean CPU time (s)	% TL reached	Mean CPU time (s)	% TL reached	Mean CPU time (s)	Mean CPU time (s)	Mean CPU time (s)
5	5.2	0	3.1	0	4.7	0	0.96	0.32	0.86
10	161.3	0	67.6	0	87.2	0	4.09	0.3	1.74
15	876.9	0	337.9	0	519.5	0	10.98	0.38	3.1
20	2593.5	0	800.6	0	1003.7	0	19.84	0.9	5.1
25	5729.4	0	2157.8	0	2218.4	0	30.67	2.18	8.28
30	6864.1	0	2108.4	0	3215.3	0	52.77	3.61	14.05
35	15037	0	3819.7	0	4412.9	0	73.34	4.34	17.34
40	TL	48.2	5858.3	0	6009.8	0	87.01	7.39	28.29
45	TL	100	8579.4	0	11490.5	0	148.23	11.82	38.83
50	TL	100	10510.5	0	15901.2	0	161.48	16.75	43.26
55	TL	100	190004.3	12.8	TL	47.3	218.51	31.27	63.88
60	TL	100	TL	80.4	TL	100	242.03	47.94	74.69
65	TL	100	TL	100	TL	100	286.38	64.24	107.36
70	TL	100	TL	100	TL	100	335.88	85	130.08

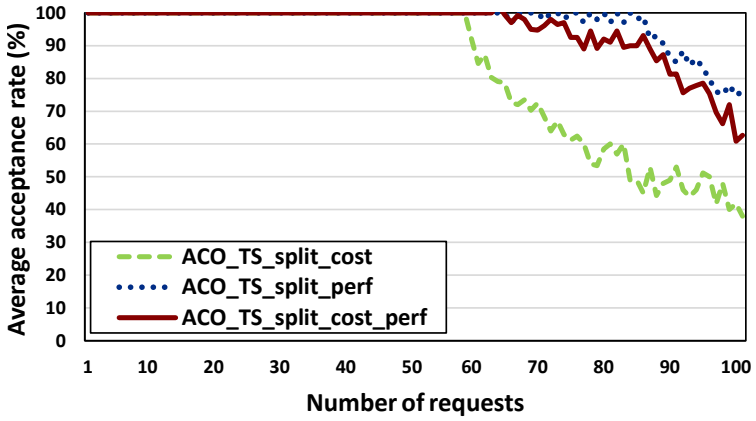

Figure 6.4 Average acceptance with *Experiment 2*

Regarding the acceptance rate, which represents one of the most conclusive metrics to evaluate the efficiency of a splitting strategy that considers the performance of VNRs, we measure it as the number of VNRs that are successfully mapped by all CPs selected by the splitting strategy, divided by the total number of incoming VNRs. In our experiment, if only one VM or one VL of a request has a failed embedding, we consider the entire VNR rejected. As shown in Fig. 6.4, the acceptance rate decreases in average according to the arrival of requests. This is mostly due to the fact that available resources of CPs can be limited over time because of requests that have not yet expired. *ACO_TS_Split_perf* and *ACO_TS_Split_cost_perf* lead to higher acceptance rates, with almost 0 rejection for the first 65 incoming VNRs and a total average respectively of about 97.29% and 95.11%. *ACO_TS_Split_cost* generates the lowest acceptance rates with a total average of about 82.88%, where the requests have 100% of acceptation only for the first 59 incomings. These results are due to the fact that *ACO_TS_Split_perf* and *ACO_TS_Split_cost_perf* take into account the performance of CPs' networks in the splitting decisions, as well as the resource capacity constraints, which improves on average the acceptance rate better than *ACO_TS_Split_cost*. Indeed, the latter is totally based on the minimization of resource prices, with no resource capacity or performance consideration. This makes this approach always selecting on average the same CPs with the same lowest resource prices, and so can easily exceed the CPs' capacities.

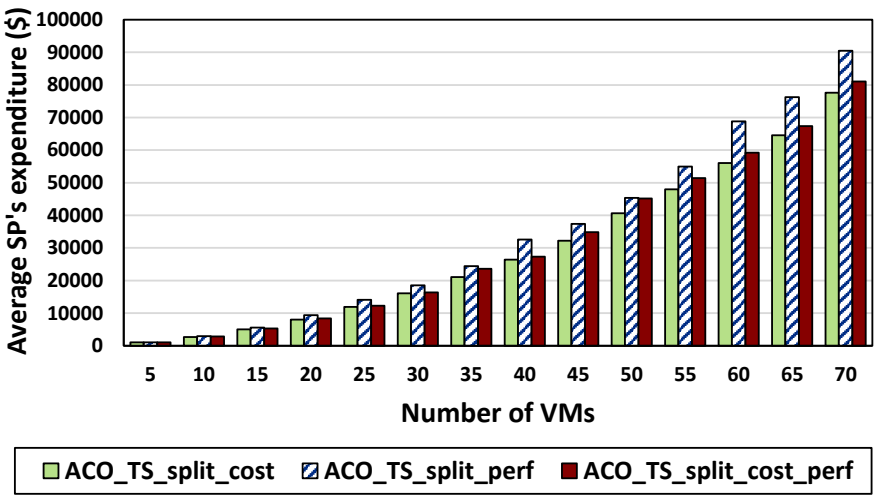

Figure 6.5 Average SP's expenditure with *Experiment 2*

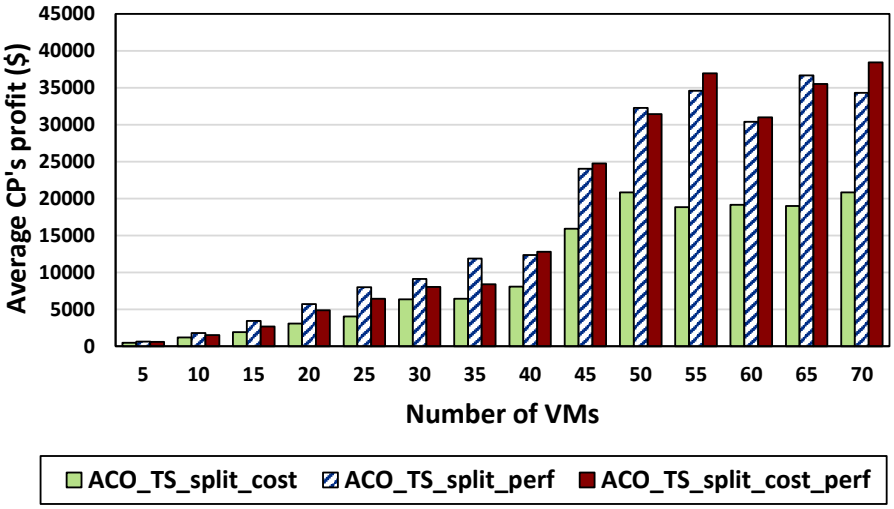

Figure 6.6 Average CP's profit with *Experiment 2*

Fig. 6.5 and Fig. 6.6 show respectively the average SP's expenditure and CP's profit per size of request. As expected, the expenditures are the lowest for the SP with *ACO_TS_Split_cost* approach, with a total average of about 29390.2 $. *ACO_TS_Split_perf*, which considers any cost aspects, generates the highest SP's expenditures, with an average of about 34413 $, compared to *ACO_TS_Split_cost_perf* approach (about 31160.9 $ of average). Considering both the acceptance rate performance metric and the cost, we can notice that *ACO_TS_Split_cost_perf* approach improves the ratio performance/cost by about 8% for the SP, compared to *ACO_TS_Split_cost* and *ACO_TS_Split_perf* approaches.

On the other hand, CPs generate more profit with *ACO_TS_Split_perf* and *ACO_TS_Split_cost_perf* (with a total average respectively of about 17512.8 $ and 17385.1 $) than with *ACO_TS_Split_cost* approach, which results in a total average of profit of about 10437.2$. This is especially due to the higher acceptance rate under *ACO_TS_Split_perf* and *ACO_TS_Split_cost_perf* approaches which can increase the revenue for a CP, and improve on average the profit of CPs by about 67.2%. This could encourage CPs to disclose more resource information to the VNP and to become more open about interoperability, in order to improve the resource utilization efficiency and therefore to increase the acceptance rate and the profit.

6.6 Conclusion

In this paper, the NP-hard optimization problem of virtual resources embedding across multiple IaaS-based cloud networks has been addressed. An efficient multicloud VNRs splitting strategy, formalized as a mathematical ILP model has been proposed, with the objective of improving the performance and the QoS of resulting VNR segments that are mapped onto the selected cloud infrastructure networks, while minimizing at the same time the resource provisioning expenditures for the SP. The proposed splitting model is first solved with the exact approach executed with small instances of the problem. Taking into account the NP-hard nature of the problem, an efficient hybrid approach based on an integrative combination of ACO and TS metaheuristics, is proposed in order to find good feasible solutions in reasonable time with large instances of the problem. Simulation results show that the proposed hybrid approach is able to generate, in a highly reduced computing time, solution costs very close to the lower bounds, with an average cost gap from the exact solution of 0% to a maximum of about 3.42%. On the other hand, the comparison results with other splitting approaches considering only the cost minimization or the performance maximization, show the efficiency of the proposed strategy model tailored to both the SP and the CP, which improves the ratio performance/cost by about 8% for the SP and the profit by about 67.2% for the CP.

Appendix A TS ALGORITHM

The proposed TS algorithm is presented in two versions. The first version is described in Fig. 6.7, which is the one used for the hybrid *ACO_TS_Split* approach, and the second version, described in Fig. 6.8, represents *TS_Split* approach.

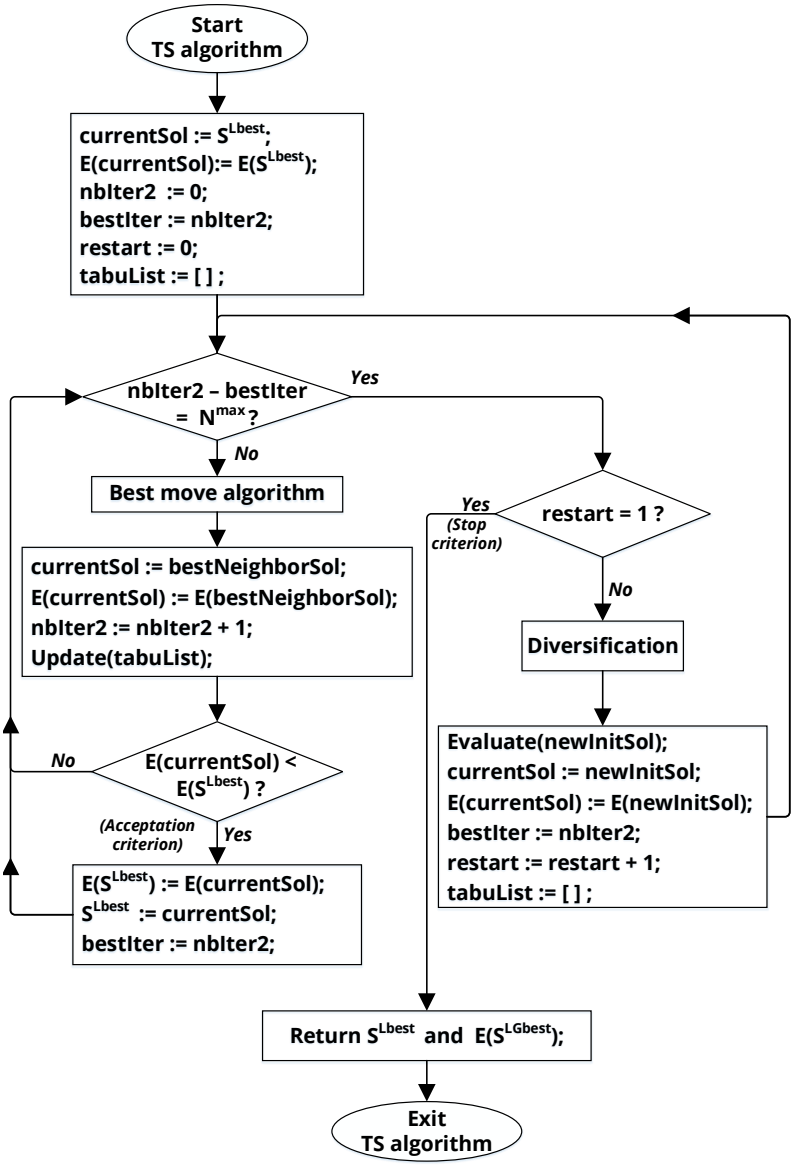

Figure 6.7 TS algorithm for *ACO_TS_Split* approach

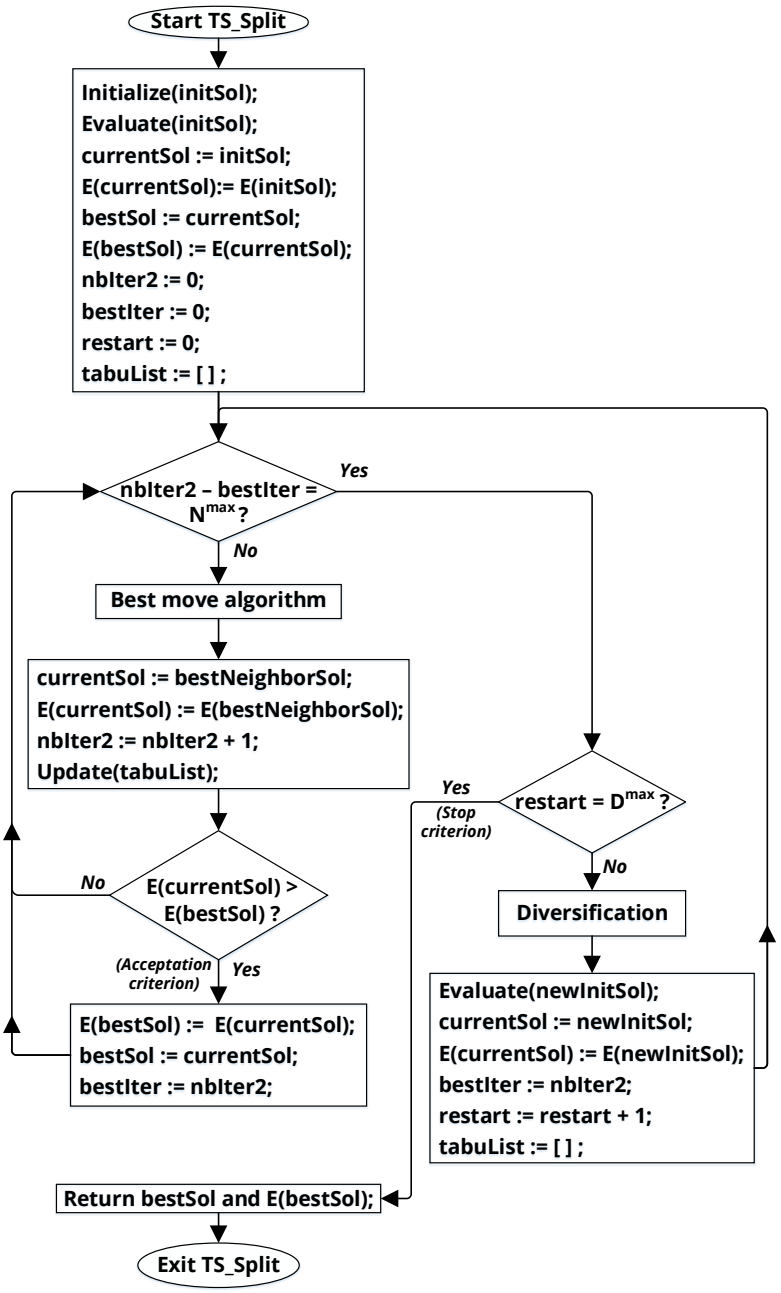

Figure 6.8 *TS_Split* algorithm

While TS with *ACO_TS_Split* starts with the best local ant's solution S^{Lbest}, *TS_Split* approach begins with an initial solution *initSol* we randomly generate, by assigning each VM to a random selected CP offering its category of node and by assigning each VL to the shortest path between a pair of CPs.

Both versions include a LS process and a long-term memory (*Diversification*) mechanism. At each iteration of the LS process, the latter tries to improve S^{Lbest} (or the best solution

bestSol in the case of *TS_Split* approach), by only accepting improving solutions (*acceptation criterion*). If S^{Lbest} (or *bestSol*) has not been improved after N^{max} iterations, the *Diversification* mechanism is then performed. The latter generates a new unexplored initial solution *newInitSol*, from which the LS process restarts the search in order to find a better solution. The *Diversification* mechanism is run for D^{max} restarts before *TS_Split* algorithm stops, but for *ACO_TS_Split* it is only run once.

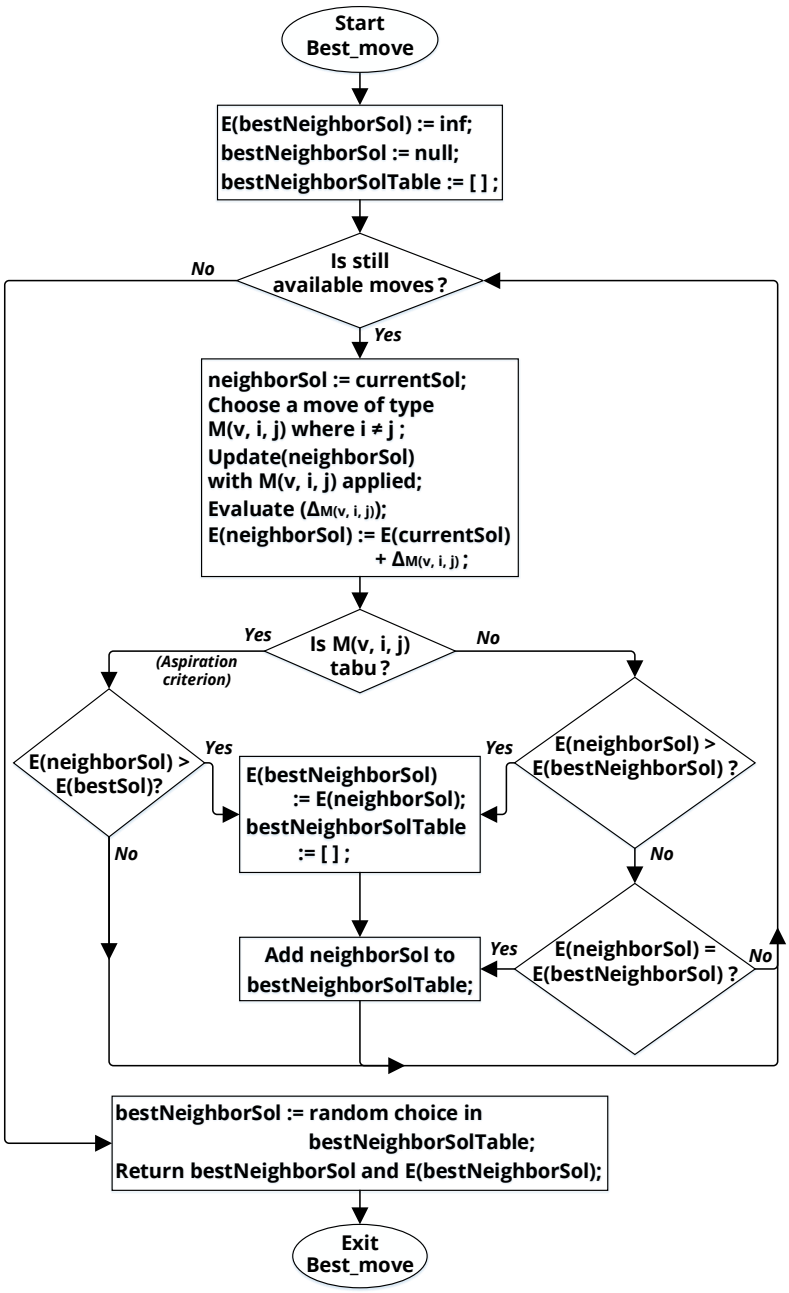

Figure 6.9 Best move algorithm

A.1 LS process

Each iteration of the LS process, in both version of the TS algorithm, consists in moving from a current solution *currentSol* to its best neighbor solution *bestNeighborSol*, by performing a *Best_move* algorithm detailed in Fig. 6.9. The latter explores all the neighborhood of *currentSol* to choose the best move to apply to *currentSol*. In our algorithm, *currentSol* moves to any neighbor solution *neighborSol* by applying a movement $M(v,i,j)$, which moves VM v from its actual assigned CP $i \in I_v^C$ in *currentSol* to a different CP $j \in I_v^C$ in *neighborSol* (and consequently all VLs related to VM v). The move $M(v,i,j)$ that generates the highest evaluation cost $E(bestNeighborSol)$, depending on its tabu status and the aspiration criterion (which allows a tabu move if only the latter improves the best solution so far), is chosen. Also, instead of selecting the first (or the last) best move after evaluating all the available moves, our algorithm considers all the moves that generate the same highest $E(bestNeighborSol)$ and randomly selects one of them.

Once the best move $M(v,i,j)$ is applied, the list of forbidden moves *tabuList* is updated by adding into it the pair (v,i), in order to forbid the heuristic to assign VM v to CP i for S^{TL} iterations, and therefore to avoid repeated solutions that can be trapped in a local minima.

Furthermore, instead of entirely evaluating at each iteration $E(neighborSol)$, which is very computationally expensive, only the evaluation cost difference $\Delta_{M(v,i,j)}$ generated by the move $M(v,i,j)$ from *currentSol* to *neighborSol* is calculated, and $E(neighborSol)$ is updated as $E(neighborSol) = E(currentSol) + \Delta_{M(v,i,j)}$. This approach decreases significantly the calculation time from the complexity order of $O(n^2)$ to $O(n)$, with n representing the number of VMs in a request.

The gain $\Delta_{M(v,i,j)}$, with $v \in N_a^V,\ a \in A,\ i,j \in I_v^C,\ j \neq i$, is defined as follows:

$$\Delta_{M(v,i,j)} = \Delta^N_{M(v,i,j)} \ + \ \Delta^{P^N}_{M(v,i,j)} \ + \ \Delta^L_{M(v,i,j)} \ + \ \Delta^{P^L}_{M(v,i,j)} \tag{A.1}$$

$\Delta^N_{M(v,i,j)}$ represents the intrinsic node cost difference from *currentSol* to *neighborSol*, given by:

$$\Delta^N_{M(v,i,j)} = \sum_{r \in R} \left(\frac{\sum\limits_{\psi \in \Gamma^V} g^S_{raj}(\psi)\, q_{rv}}{w_{rv} C^N_{raj}} - \frac{\sum\limits_{\psi \in \Gamma^V} g^S_{rai}(\psi)\, q_{rv}}{w_{rv} C^N_{rai}} \right) \tag{A.2}$$

$\Delta^{P^N}_{M(v,i,j)}$ represents the node penalty cost difference from *currentSol* to *neighborSol*, given by:

$$\Delta^{P^N}_{M(v,i,j)} = -10^9 \times \sum_{r \in R} \Big(\min\left(0, Q_{raj} - (Q^*_{raj} + q_{rv})\right) + \min\left(0, Q_{rai} - (Q^*_{rai} - q_{rv})\right) \Big) \tag{A.3}$$

$\Delta^L_{M(v,i,j)}$ represents the intrinsic link cost difference from *currentSol* to *neighborSol*, given by:

$$\Delta^L_{M(v,i,j)} = \sum_{\psi \in \Gamma^V} \left(\begin{array}{c} \sum\limits_{\substack{l_1 = vu \\ u \in N^V \\ u \neq v}} \left(\frac{g^L_{t_1 \varphi^1_{jm}(l_1)}(\psi)\, b_{l_1}}{C^L_{t_1 \varphi^1_{jm}(l_1)}} - \frac{g^L_{t_1 \varphi^1_{im}(l_1)}(\psi)\, b_{l_1}}{C^L_{t_1 \varphi^1_{im}(l_1)}} \right) \\ + \\ \sum\limits_{\substack{l_2 = uv \\ u \in N^V \\ u \neq v}} \left(\frac{g^L_{t_2 \varphi^1_{mj}(l_2)}(\psi)\, b_{l_2}}{C^L_{t_2 \varphi^1_{mj}(l_2)}} - \frac{g^L_{t_2 \varphi^1_{mi}(l_2)}(\psi)\, b_{l_2}}{C^L_{t_2 \varphi^1_{mi}(l_2)}} \right) \end{array} \right) \tag{A.4}$$

where $\varphi^1_{jm}(l_1)$ (respectively $\varphi^1_{im}(l_1)$) represent respectively the new and the old assignment of VL $l_1 = vu \in L^V_{t_1}$, and $\varphi^1_{mj}(l_2)$ (respectively $\varphi^1_{mi}(l_2)$) represent respectively the new and the old assignment of VL $l = uv \in L^V_{t_1}$. u represents each communicating VM with v and m is the CP to which VM u is assigned.

$\Delta^{P^L}_{M(v,i,j)}$ represents the link penalty cost difference from *currentSol* to *neighborSol*, given by:

$$\Delta^{P^L}_{M(v,i,j)} = 10^9 \times \left(\left| H^L_{neighborSol} \right| - \left| H^L_{currentSol} \right| \right) \tag{A.5}$$

where $H^L_{S_k \oplus (v,i)}$ represents the set of all VLs connected with v whose the maximum delay is violated in $S_k \oplus (v,i)$, given by:

where $H^L_{currentSol}$ and $H^L_{neighborSol}$ represent the set of VLs whose the resulting assignment, respectively in the current solution and the neighbor solution, violates the maximum delay allowed on the links.

A.2 Long-term memory mechanism

The *Diversification* mechanism helps to generate new solutions from so far unexplored regions. To this end, after each iteration of the LS process, the resulting solution is kept in a statistics memory table, which stores, from the beginning of the algorithm, the number of times each VM has been assigned to a CP. Then, based on the least explored assignment criterion, the long-term memory mechanism creates a new initial solution *newInitSol* from which a new exploration restarts.

A.3 Parameters of TS algorithm

The preliminary tests allowed us to define the optimal parameters of the TS algorithm. For both versions, the length S^{TL} is defined as $1/3 * \sum_{a \in A} \left|N_a^V\right| * |I_a|$, where I_a represents the set of CPs supporting nodes of category $a \in A$. N^{max} for *ACO_TS_Split* approach is defined in the main manuscript. For approach, the parameters N^{max} and D^{max} are defined as follows: for small sized VNRs, $N^{max} = 40 * \sqrt{|N^V| * |I|}$ and $D^{max} = 25 * \sqrt{|N^V| + |I|}$. For large sized VNRs, $N^{max} = 20 * \sqrt{|N^V| * |I| / 2}$ and $D^{max} = 5 * 10^3 * \sqrt{|I| / |N^V|^2}$.

Appendix B FORMULATION OF THE VNR SEGMENTS INTRACLOUD MAPPING PROBLEM

Here, we present the formulation of the intracloud mapping approach we use in our framework to resolve the mapping phase problem. All the notation used can be consulted in Table 6.7. Details related to the setup of cost parameters used for our simulations are also given.

Table 6.7 Notation for the VNR segments intracloud mapping problem

Symbols	Description
	Global sets
A	Set of node categories
T	Set of link types
R	Set of computational resource types
I	Set of CPs
	Virtual network request segment
G_i^V	Graph representing the VNR segment assigned to CP $i \in I$
N_i^V	Set of VMs assigned to CP $i \in I$
L_i^V	Set of virtual links assigned to CP $i \in I$
N_{ai}^V	Set of VMs of node category $a \in A$ assigned to CP $i \in I$
L_{ti}^V	Set of virtual links of type $t \in T$ exclusively assigned to CP $i \in I$
$L_{t\phi_i}^V$	Set of virtual links of type $t \in T$ assigned to an intercloud link whose one of the endpoints is CP $i \in I$
q_{rv}	Amount of resource $r \in R$ required by VM $v \in N_i^V$
b_l	Bandwidth demand of virtual link $l \in L_i^V$
δ_l	Maximum delay allowed for virtual link $l \in L_i^V$
Γ^V	Lifetime of a VNR segment

Table 6.7 Notation for the VNR segments intracloud mapping problem

	Substrate intracloud network
G_i^S	Graph representing the substrate network of CP $i \in I$
N_i^S	Set of substrate nodes in G_i^S
L_i^S	Set of substrate links in G_i^S
D_i	Set of data centers of CP $i \in I$
ϕ_i	PoP node (transit network) of CP $i \in I$
$\mathcal{F}_i$	Set of all paths in CP i's network
$\mathcal{P}_{dc}$	Set of all paths between data centers d and c, $d, c \in D_i$
$\mathcal{P}_{d\phi_i}$	Set of all paths between data center $d \in D_i$ and PoP node ϕ_i
$\mathcal{K}_e$	Set of all paths in $\mathcal{F}_i$ spanning link $e \in L_i^S$
B_{te}	Available bandwidth capacity of the channel of type $t \in T$ of link $e \in L_i^S$
d_e	Delay bound on link $e \in L_i^S$
Q_{rad}	Available capacity of resource $r \in R$ in data center $d \in D_i$ for nodes of category $a \in A$ ($Q_{rad} \in \mathbb{N}_1$)
U_{adr}	Usage of resource $r \in R$ by all VMs of node category $a \in A$ assigned to data center $d \in D_i$
E_{adr}	Average power (in watts) consumed by a VM of node category $a \in A$ assigned to data center $d \in D_i$, in terms of resource $r \in R$
ω_d	Average power (in watts) consumed by all VMs assigned to data center $d \in D_i$
ρ_d	Power Usage Effectiveness (PUE) of data center $d \in D_i$
θ_d	CO_2 emissions in data center $d \in D_i$ (in g/KWh)
	Costs
$g_{rai}^N(\psi)$	Unit price of resource $r \in R$ supplied by CP $i \in I$ for nodes of category $a \in A$, during the time period ψ
$g_{ti}^L(\psi)$	Unit bandwidth price supplied by CP $i \in I$ for links oftype $t \in T$, during the time period ψ
$g_{t\phi_i}^L(\psi)$	Unit extra bandwidth price for transiting through PoP network ϕ_i for links of type $t \in T$, during the time period ψ
$c_{rai}^{\mathcal{S}}$	Unit resource cost for CP $i \in I$ for using resource $r \in R$ for nodes of category $a \in A$ (in \$/unit)
c_{ti}	Unit bandwidth cost for CP $i \in I$ for using a link channel of type $t \in T$ (in \$/Mbps)
$c_{ti}^{\mathcal{E}}$	Extra unit bandwidth cost for CP $i \in I$ for using a link channel of type $t \in T$ (in \$/Mbps)
c_d^{ω}	Unit electricity cost in data center $d \in D_i$ (in \$/MWh)
$\wp^{\mathcal{D}}$	Penalty for each millisecond of delay on virtual links (in \$/ms)
$\wp_d^{\mathcal{O}}$	Penalty for emitting CO_2 in data center $d \in D_i$ (in \$/tonne)
$C_i^{\mathcal{S}}$	Total computing resource cost for CP $i \in I$ (in \$/h)
$C_i^{\mathcal{T}}$	Total traffic cost for CP $i \in I$ (in \$/h)
$C_i^{\mathcal{D}}$	Total delay penalty for CP $i \in I$ (in \$/h)
C_i^{ω}	Total energy cost for CP $i \in I$ (in \$/h)
$C_i^{\mathcal{O}}$	Total environmental penalty for CP $i \in I$ (in \$/h)
	Decision variables
X_{vn}	Binary variable set to **1** if VM $v \in N_{ai}^V$, $a \in A$, is assigned to substrate node $n \in N_i^S$; **0** otherwise
$Y_{l\varphi}$	Binary variable set to **1** if virtual link $l \in L_{ti}^V$, $t \in T$, is assigned to path $\varphi \in \mathcal{P}_{dc}$, $d, c \in D_i$; **0** otherwise
$Z_{k\gamma}$	Binary variable set to **1** if virtual link $k \in L_{t\phi_i}^V$, $t \in T$, is assigned to path $\gamma \in \mathcal{P}_{d\phi_i}$, $d \in D_i$; **0** otherwise

B.1 Adopted intracloud mapping approach

In our framework, we resolve the mapping phase problem by evaluating, for each selected CP i, the total revenue $\mathcal{R}_i$ and the total cost $\mathcal{C}_i$ of embedding, in order to maximize the profit P_i as follows:

$$P_i \;=\; \mathcal{R}_i \;-\; \mathcal{C}_i \qquad\qquad \forall\, i \in I \tag{B.1}$$

where

$$\begin{aligned}\mathcal{R}_i = \;& \sum_{d\in D_i}\sum_{a\in A}\sum_{v\in N_{ai}^V}\sum_{r\in R}\sum_{\psi\in\Gamma^V} g_{rai}^N(\psi)\, q_{rv} X_{vd} \;+ \\ & \sum_{d\in D_i}\sum_{c\in D_i}\sum_{\varphi\in\mathcal{P}_{dc}}\sum_{t\in T}\sum_{l\in L_{ti}^V}\sum_{\psi\in\Gamma^V} g_{ti}^L(\psi)\, b_l Y_{l\varphi} \;+ \\ & \sum_{d\in D_i}\sum_{\gamma\in\mathcal{P}_{d\phi_i}}\sum_{t\in T}\sum_{k\in L_{t\phi_i}^V} \left(g_{ti}^L(\psi) + g_{t\phi_i}^L(\psi)\right) b_k Z_{k\gamma}\end{aligned} \tag{B.2}$$

and

$$\mathcal{C}_i = \sum_{\psi\in\Gamma^V}\Big(\;\alpha\mathcal{C}_i^{\mathcal{S}} \;+\; \beta\mathcal{C}_i^{\mathcal{T}} \;+\; \lambda\mathcal{C}_i^{\mathcal{D}} \;+\; \varpi\mathcal{C}_i^{\omega} \;+\; o\mathcal{C}_i^{\mathcal{O}}\Big) \tag{B.3}$$

with each cost weighted by a parameter that allows CPs to modify the costs priority.

The following are the formulas that define each cost and penalty of the total cost $\mathcal{C}_i$.

Servers cost is re-stated as computational resources cost and is calculated as follows:

$$C_i^{\mathcal{S}} = \sum_{d\in D_i}\sum_{r\in R}\sum_{a\in A}\sum_{v\in N_{ai}^V} c_{rai}^{\mathcal{S}} q_{rv} X_{vd} \tag{B.4}$$

The total traffic cost for CP i is defined as follows, with an extra cost stated for intercloud virtual links:

$$\begin{aligned}C_i^{\mathcal{T}} = \;& \sum_{d\in D_i}\sum_{c\in D_i}\sum_{\varphi\in\mathcal{P}_{dc}}\sum_{e\in\varphi}\sum_{t\in T}\sum_{l\in L_{ti}^V} c_{ti} b_l Y_{l\varphi} \\ & + \sum_{d\in D_i}\sum_{\gamma\in\mathcal{P}_{d\phi_i}}\sum_{e\in\gamma}\sum_{t\in T}\sum_{k\in L_{t\phi_i}^V} \left(c_{ti} + c_{ti}^{\mathcal{E}}\right) b_k Z_{k\gamma}\end{aligned} \tag{B.5}$$

The total delay penalty for CP i is re-defined as follows:

$$C_i^{\mathcal{D}} = \wp^{\mathcal{D}} \left(\begin{array}{l} \displaystyle\sum_{d\in D_i}\sum_{c\in D_i}\sum_{\varphi\in\mathcal{P}_{dc}}\sum_{e\in\varphi}\sum_{t\in T}\sum_{l\in L_{ti}^V} d_e Y_{l\varphi} \\ \displaystyle +\sum_{d\in D_i}\sum_{\gamma\in\mathcal{P}_{d\phi_i}}\sum_{e\in\gamma}\sum_{t\in T}\sum_{k\in L_{t\phi_i}^V} d_e Z_{k\gamma} \end{array} \right) \tag{B.6}$$

The average power consumed by all VMs assigned to data center $d \in D_i$ is re-stated as follows, estimated as a function of the utilization of CPU resource and disk resource, by following the principles used in [24]:

$$\omega_d = \sum_{a \in A} \left(E_{dar_1} U_{dar_1} + E_{dar_2} U_{dar_2} \right), \quad \forall\ d \in D_i, \tag{B.7}$$

where E_{dar_1} and E_{dar_2} represent (in watts) the average power consumed by a VM of node category $a \in A$ assigned to data center d, respectively in terms of CPU resource and disk resource. U_{dar_1} and U_{dar_2} define respectively the CPU resource usage and the disk resource usage by all VMs of node category $a \in A$ assigned to data center d, given by:

$$U_{dar} = \frac{\sum\limits_{v \in N_{ai}^V} q_{rv} X_{vd}}{\max\left(Q_{rad}, 1\right)}, \qquad \forall\ d \in D_i,\ a \in A,\ r \in R \tag{B.8}$$

The total energy cost for CP i is then defined as follows:

$$C_i^{\omega} = 10^{-6} \sum_{d \in D_i} c_d^{\omega} \rho_d \omega_d, \tag{B.9}$$

The total environmental penalty for CP i is given as follows:

$$C_i^{\mathcal{O}} = 10^{-9} \sum_{d \in D_i} \wp_d^{\mathcal{O}} \theta_d \rho_d \omega_d \tag{B.10}$$

The objective function for the intracloud mapping phase is then defined as follows:

MAX

$$P_i \tag{B.11}$$

Subject to:

$$\sum_{n \in N_i^S \setminus D_i} X_{vn} = 0, \qquad \forall\ v \in N_{ai}^V,\ a \in A \tag{B.12}$$

$$\sum_{d \in D_i} X_{vd} \leq 1, \qquad \forall\ v \in N_{ai}^V,\ a \in A \tag{B.13}$$

Constraints (B.12) and (B.13) ensure respectively that VMs are only assigned to data centers, and each VM should be assigned to exactly one data center.

$$Y_{l\varphi} \leq \frac{X_{ud} + X_{vc}}{2}, \qquad \forall\ l = uv \in L_{ti}^{V},\ t \in T,\ \ u, v \in N_{ai}^{V}(v \neq u),\ a \in A, \\ \varphi \in \mathcal{P}_{dc},\ d, c \in D_i \tag{B.14}$$

$$Z_{k\gamma} \leq X_{vd}, \qquad \forall\ k = v\phi_i \in L_{t\phi_i}^{V},\ t \in T,\ v \in N_{ai}^{V},\ \ \gamma \in \mathcal{P}_{d\phi_i},\ d \in D_i \tag{B.15}$$

Constraints (B.14) and (B.15) state respectively the binary value of variables $Y_{l\varphi}$ and $Z_{k\gamma}$.

$$\sum_{d \in D_i} \sum_{c \in D_i} \sum_{\varphi \in \mathcal{P}_{dc}} Y_{l\varphi} \leq 1, \qquad \forall\ l \in L_{ti}^{V},\ t \in T \tag{B.16}$$

$$\sum_{d \in D_i} \sum_{\gamma \in \mathcal{P}_{d\phi_i}} Z_{k\gamma} \leq 1, \qquad \forall\ k \in L_{t\phi_i}^{V},\ t \in T \tag{B.17}$$

Constraint (B.16) ensures that each virtual link exclusively assigned to a CP should be mapped to a unique path between two different data centers, otherwise to a unique path intra-data center. Constraint (B.17) ensures that each virtual link partially assigned to a CP should be mapped to a unique path between a data center and the PoP node of the CP.

$$\sum_{v \in N_{ai}^{V}} q_{rv} X_{vd} \leq Q_{rad}, \qquad \forall\ r \in R,\ a \in A,\ d \in D_i \tag{B.18}$$

$$Q_{rad} = Q_{rad} - \sum_{v \in N_{ai}^{V}} q_{rv} X_{vd}, \qquad \forall\ r \in R,\ a \in A,\ d \in D_i \tag{B.19}$$

Constraint (B.18) ensures that the total amount of a resource required by all VMs assigned to a data center must not exceed its residual capacity. Constraint (B.19) updates this residual capacity after each VNR is successfully mapped.

$$B_{\overrightarrow{te}} = B_{\overleftarrow{te}}, \qquad \forall\ e \in L_{ti}^{S} \tag{B.20}$$

Constraint (B.20) states that the bandwidth capacity of a channel of a link type is the same in both directions on every substrate link.

$$\sum_{m \in N_i^S} f_{nm}^{uv} + X_{vn} b_l = \sum_{m \in N_i^S} f_{mn}^{uv} + X_{un} b_l, \\ \forall u, v \in N_i^V,\ v \neq u, n \in N_i^S,\ l = uv \in L_i^V \tag{B.21}$$

Constraint (B.21) guaranties the flow conservation for every amount of traffic $l = uv$ from VM u to VM v, with a bandwidth demand b_l routed on substrate link $e = mn \in L_i^S$.

$$\sum_{\varphi \in \mathcal{K}_e} \sum_{l \in L_{ti}^{V}} b_l Y_{l\varphi} + \sum_{\gamma \in \mathcal{K}_e} \sum_{k \in L_{t\phi_i}^{V}} b_k Z_{k\gamma} \leq B_{te}, \forall\ e \in L_i^S,\ t \in T \tag{B.22}$$

$$B_e = B_e - \left(\sum_{\varphi \in \mathcal{K}_e} \sum_{l \in L_{it}^V} b_l Y_{l\varphi} + \sum_{\gamma \in \mathcal{K}_e} \sum_{k \in L_{t\phi_i}^V} b_k Z_{k\gamma} \right),\ \forall\ e \in L_{ti}^S,\ t \in T \tag{B.23}$$

Constraint (B.22) ensures that the total bandwidth demand of all virtual links routed on a substrate link must not exceed the residual bandwidth capacity of the corresponding channel. Constraint (B.23) updates this residual capacity after a VNR is successfully mapped.

$$\sum_{d \in D_i} \sum_{c \in D_i} \sum_{\varphi \in \mathcal{P}_{dc}} \sum_{e \in \varphi} d_e Y_{l\varphi} \leq \delta_l,\ \forall\ l \in L_{ti}^V,\ t \in T \tag{B.24}$$

$$\sum_{d \in D_i} \sum_{\gamma \in \mathcal{P}_{d\phi_i}} \sum_{e \in \gamma} d_e Z_{l\gamma} \leq \delta_k,\ \ \forall\ k \in L_{t\phi_i}^V,\ t \in T \tag{B.25}$$

Constraints (B.24) and (B.25) ensure that the restriction on the maximum delay allowed for a virtual link is not violated.

$$X_{vn} \in \{0,1\} \qquad \forall\ v \in N_{ai}^V,\ a \in A,\ n \in N_i^S \tag{B.26}$$

$$Y_{l\varphi} \in \{0,1\} \quad \forall\ l \in L_{ti}^V,\ t \in T,\ \varphi \in \mathcal{P}_{dc},\ d, c \in D_i \tag{B.27}$$

$$Z_{k\gamma} \in \{0,1\} \quad \forall\ k \in L_{t\phi_i}^V,\ t \in T,\ \gamma \in \mathcal{P}_{d\phi_i},\ d \in D_i \tag{B.28}$$

$$\omega_d \in \mathbb{R}_{\geq 0} \qquad \forall\ d \in D_i \tag{B.29}$$

Constraints (B.26) - (B.29) express the domain of definition of each variable.

B.2 Cost parameters setup

Most of cost parameters setting follows the experimentation setup defined in [20], but they are specified for an hour time period. We chose the delay as the first optimization priority of the multi-objective function, by using the same weight attribution as in [20]. The unit computational resource cost is randomly determined in the interval \$[1, 5]/core/h for CPU resource, \$([1, 5]/10)/GB/h for memory resource and \$([1, 5]/100)/GB/h for disk resource. The unit bandwidth cost for each link type is randomly chosen in the interval \$([5, 10]/1000)/Mbps/h, and the corresponding extra unit bandwidth cost is estimated at 25% of the initial unit cost. The delay penalty is set to \$1.15/(ms/packet)/h. The average power consumed by a VM assigned to a data center is randomly set in the interval [200 300] W (with 60% of the energy for CPU resource and 40% for disk resource). The unit electricity cost is randomly distributed in the interval \$[30 70]/MWh. The PUE of a data center is 1.5. The amount of CO_2 emissions in a data center is calculated by summing values in the set {10 66 443 960} g/KWh, each of them first multiplied by a greenness factor randomly set between 0 and 1. The penalty for emitting CO_2 in a data center is defined as \$1000/tonne.

CHAPITRE 7 DISCUSSION GÉNÉRALE

Dans ce chapitre, nous effectuerons une analyse critique de l'ensemble des travaux de recherche réalisés dans le cadre de cette thèse. Nous discuterons d'abord des démarches méthodologiques adoptées en vue d'atteindre les différents objectifs fixés. Ensuite, une analyse des résultats issus de nos travaux sera dressée, afin de mieux mettre en lumière la portée de ces derniers à travers les contributions de cette thèse.

7.1 Analyse des démarches méthodologiques

L'objectif principal de cette thèse consistait principalement à concevoir un cadre hiérarchique d'intégration des requêtes de réseaux virtuels (VNRs) dans un environnement multiCloud offrant des services sous le modèle IaaS. Les travaux de recherche présentés dans cette thèse ont alors débuté par une revue de littérature. Cette dernière nous a permis de faire état des modèles et approches de résolution proposés dans les travaux existants pour un tel problème d'optimisation, et d'en dégager les limites et failles. Sur cette base, nos travaux ont donné lieu à des propositions organisées en différents volets et mises en œuvre dans quatre articles scientifiques, dont trois en particulier ont fait l'objet de chapitres dans ce manuscrit.

Dans le premier volet de la thèse, nous avons développé une stratégie de partitionnement des VNRs à travers les différentes infrastructures de l'environnement multCloud sous-jacent. La stratégie proposée, formalisée dans un modèle mathématique linéaire et résolue avec une méthode exacte, vise particulièrement à optimiser les performances des différentes applications à héberger et à maximiser leur taux d'acceptation, sous les contraintes de Qualité de Service (QdS) imposées dans les VNRs. Un modèle mathématique a également été adopté afin de résoudre la phase d'hébergement intraCloud des partitions de VNRs, dans le but notamment de pouvoir évaluer et valider le modèle stratégique de partitionnement proposé selon plusieurs critères de performance.

Par la suite, afin de pallier la question de l'explosion combinatoire induite par la résolution avec la méthode exacte d'instances de grande taille du problème, dû à la nature NP-Difficile de ce dernier, une méthode approximative de résolution est alors conçue dans le deuxième volet. Cette dernière constitue précisément en une adaptation de la métaheuristique de Recherche Taboue (TS) à notre problème, afin de trouver des solutions (presque) optimales, mais en un temps de calcul suffisamment raisonnable.

Dans le dernier volet, nous avons proposé une extension du modèle mathématique de partitionnement des VNRs, afin de pouvoir minimiser également les dépenses du SP relatives à l'approvisionnement des ressources et des services Cloud. Le nouveau modèle mathématique est d'abord résolu de manière exacte sur des instances de petite taille du problème. Ensuite, pour être à même de traiter des instances de plus grande taille, une approche approximative hybride, basée sur une combinaison intégrative des métaheuristiques de colonies de fourmis (ACO) et de TS, est alors développée, permettant de résoudre avec davantage d'efficacité le nouveau modèle proposé.

Dans la suite, une analyse de la méthodologie adoptée pour réaliser les divers travaux est effectuée, afin de discuter des différents choix opérés par rapport à d'autres alternatives qui pouvaient être envisageables. Cette analyse portera principalement sur la modélisation proposée du problème dans sa globalité, les différentes approches de résolution mises en œuvre, ainsi que sur l'environnement et les outils de simulation utilisés pour l'évaluation de performance des modèles conçus.

7.1.1 Modélisation du problème global du VNE multiCloud

Le but principal dans la résolution du problème du VNE dans un contexte multiCloud était de notamment proposer un nouveau modèle d'optimisation du problème permettant de combler les limites et failles autour des travaux existants sur le sujet, présentés à la Section 2.5. Concevoir un tel modèle dans un environnement semblable nécessitait d'intégrer, dans la modélisation du problème global, différents facteurs clés liés aux VNRs et aux domaines multiCloud et intracloud, mais également de faire abstraction sur certaines considérations techniques présentées ci-dessous, afin de pouvoir s'aligner adéquatement à la visée de cette thèse.

7.1.1.1 Éléments techniques des VNRs

Lors de la soumission d'une VNR, le SP définit les besoins sur les applications à héberger, ainsi que les caractéristiques et contraintes associées. Dans la modélisation du problème du partitionnement des VNRs, nous avons intégré ces éléments liés non seulement aux VMs mais également au trafic inter-VMs, à savoir les attributs fonctionnels et besoins en performance des VMs, le type des communications inter-VMs, la quantité de bande passante nécessaire ainsi que le délai maximal autorisé sur ceux-ci. Cependant, pour des fins de simplicité de la procédure, nous avons supposé une étape antérieure à notre travail, qui étudie les caractéristiques des VMs soumises et classifie ces dernières dans une certaine catégorie, laquelle regroupe tous les attributs fonctionnels d'une VM, incluant les contraintes de localisation,

de sécurité et de territorialité, et qui sera par la suite considérée comme paramètre d'entrée dans notre modèle d'optimisation.

Aussi, nous avons supposé que les besoins en ressources des VMs et VLs sont connues et restent invariantes durant tout le processus de VNE et pendant toute la durée de vie d'une requête. Cela nous permet de faire fi de mécanismes de re-partitionnements des VNRs existantes ou en cours de traitement au niveau de la phase de sélection des CPs, de ré-hébergements ou de migrations de VMs au niveau des phases subséquentes de traitement intraCloud, en général dans le but de pouvoir assurer que les requêtes demeurent satisfaites. En effet, l'objectif principal était plutôt de mettre en évidence la contribution de notre travail sur la même base de comparaison que les travaux existants avec lesquels nous nous comparons, et qui n'ont pas eu à faire de telles considérations au niveau de la phase de partitionnement des VNRs.

Par ailleurs, en plus du trafic inter-VMs considéré dans notre modèle d'optimisation du VNE, nous aurions pu naturellement intégrer dans la modélisation le trafic non négligeable entre les VMs et les clients finaux du SP, et ce selon les mêmes critères que ceux considérés pour le trafic inter-VMs. Cela rajouterait certes une complexité dans le temps de résolution du modèle, mais ne constituerait pas un défi supplémentaire pour ce dernier.

De plus, nous n'avons pas imposé, dans la modélisation des VNRs, des contraintes de co-hébergement dans le même DC, donc chez le même CP, pour des VMs encapsulant la même application ou échangeant du trafic, comme c'est le cas dans certains travaux (Justafort *et al.*, 2015). D'autres ont plutôt considéré des contraintes de distance maximale entre la localisation de deux VMs (Li *et al.*, 2016; Pyoung et Baek, 2018). Cette dernière considération nous semblait plus convenable pour le cadre de VNE proposé, et ainsi, dans notre modélisation, nous avons traduit cette distance en termes de délai maximal, ce critère étant directement lié à la performance des applications. De ce fait, notre modèle de partitionnement, outre de satisfaire le SP en minimisant le délai global de communication inter-VMs dans les deux phases du VNE, s'assure conjointement à ne pas violer cette contrainte de délai maximal imposée sur les VLs.

7.1.1.2 Éléments techniques de l'environnement sous-jacent

Les éléments techniques de l'environnement multiCloud, intégrés dans la modélisation du problème du partitionnement des VNRs, se basent sur le niveau d'accès du VNP sur la topologie de haut niveau visible par ce dernier, ainsi que sur les informations contenues dans le référentiel de données. Ces éléments incluent l'interconnexion entre les CPs, l'ensemble des chemins existant entre les CPs, qui sont considérés comme paramètres d'entrée à notre modèle de partitionnement, le délai moyen de propagation sur ces chemins, le nombre de

domaines PoPs que les chemins traversent, le délai moyen pour transiter par ces PoPs, ainsi que les données énoncées à la Section 3.2.1.1 et les coûts unitaires des ressources, publiés par période de temps.

Étant donné que la topologie interne détaillée des réseaux PoPs n'est pas accessible par le VNP, ces derniers ont dû être modélisés à un très haut niveau d'abstraction, où les nœuds et les liens d'appairage sont agrégés et représentés comme un réseau unique, associé d'un délai moyen pour y transiter. Par souci de simplicité, nous n'avons considéré qu'un seul réseau PoP par CP, dès lors que cela n'entrave pas la validité de notre modèle car le contraire serait également facilement adaptable à ce dernier. De même, le réseau dorsal interconnectant les différents domaines Cloud, qui peut également être constitué de différents réseaux physiques intermédiaires, n'est pas contrôlable par le VNP. De ce fait, ces derniers sont aussi représentés, à un haut niveau d'abstraction, comme des liaisons interCloud associées d'un délai moyen garanti de routage.

Aussi, vu que le VNP n'est pas en mesure de gérer la capacité de bande passante résiduelle sur les chemins interCloud, ou dans les réseaux PoPs, nous n'avons tenu en compte cette ressource que dans la phase d'hébergement intraCloud, dont le CP a par contre une connaissance complète, et non durant la phase de partitionnement. Pour les mêmes raisons, dans ce travail, les notions de fractionnement d'un même trafic entre plusieurs chemins n'était pas envisageable lors de la sélection des CPs.

Au niveau de la modélisation du réseau intraCloud, l'ensemble des éléments considérés dans le modèle adopté de (Larumbe et Sanso, 2013) ont été modélisés, incluant le réseau dorsal interconnectant les différents DCs, tout en considérant en plus l'hétérogénéité de l'infrastructure. Cependant, la structure interne des DCs n'a pas été incluse dans notre modélisation, à savoir les serveurs qui y sont logés et leurs interconnexion par les châssis et les commutateurs, etc. La résolution du problème de l'hébergement intraCloud des segments de VNRs est donc réalisée au niveau de la sélection des DCs, et non à un niveau encore plus bas de la hiérarchie, faisant référence à la sélection des serveurs physiques internes. En effet, les principales contributions de cette thèse se rapportant plus au niveau de la phase de partitionnement multiCloud des VNRs, ce niveau de granularité intraCloud considéré nous semblait suffisant pour être à même d'évaluer la stratégie de partitionnement proposée sans entraver la validité de cette dernière, dès lors que pour tous les modèles de partitionnement de la littérature avec lesquels nous nous comparons, nous avons utilisé cette même démarche méthodologique pour résoudre la phase d'hébergement intraCloud.

7.1.1.3 Modèles d'optimisation du problème global

Dans le modèle d'optimisation du problème du partitionnement des VNRs, outre d'avoir intégré les différentes considérations techniques décrites précédemment, nous avons choisi d'optimiser précisément la performance et la QdS des applications à héberger pour le SP, ainsi que les coûts monétaires induits. D'autres modèles semblables à celui de (Ardagna *et al.*, 2017), optimisant autant les coûts d'approvisionnement pour le SP que les profits des différents CPs dans la stratégie de sélection des CPs, auraient pu être envisagés également. Seulement, nous avons opté, dans le cadre de cette thèse, de n'optimiser que des critères de performance et d'économie portant intérêt au SP durant la phase de partitionnement des VNRs, en prenant tout de même le soin d'adopter un modèle d'optimisation du problème de l'hébergement intraCloud qui puisse refléter au mieux la réalité, même si cela pourrait être conflictuel aux critères optimisés dans la phase de partitionnement. En effet, tel qu'observé au niveau de la revue de littérature, plusieurs approches de la phase d'hébergement intraCloud s'offraient à nous. Cependant, notre choix s'est porté sur celle de (Larumbe et Sanso, 2013), car l'approche nous permet de considérer conjointement différents facteurs clés optimisés par le CP et traduits dans une même unité de coût (dollars) dans la fonction objectif, à savoir l'utilisation des ressources informatiques et réseautiques, l'équilibrage des charges dans l'infrastructure, l'énergie et l'impact environnemental, ainsi que le délai global de routage dans le réseau dorsal. De plus, afin de refléter davantage la réalité, nous avons intégré dans le modèle la maximisation des profits pour les CPs, des aspects qui n'étaient pas considérés dans le modèle initial. Cela nous a permis de fournir un modèle global d'optimisation du problème du VNE multiCloud, où les intérêts économiques divergents des deux parties intervenants ont été pris en compte, mais à différents niveaux de la modélisation du problème.

7.1.2 Approches de résolution et environnement de simulation

La conception des différents modèles d'optimisation étant faite, nous nous sommes alors d'abord assurés que tous les aspects considérés du problème global ont été validement intégrés, en résolvant manuellement de très petites instances du problème, autant pour la phase de partitionnement que celle de l'hébergement intraCloud. Par la suite, plusieurs alternatives de résolution plus automatique se présentaient à nous. Afin de souligner notamment la contribution de notre travail, nous avons alors opté, dans un premier temps, pour une approche de résolution optimale d'instances de taille raisonnable. Ensuite, pour des instances de taille plus importante, nous nous sommes orientés vers les approches d'approximation de solutions.

7.1.2.1 Approche de résolution exacte

La résolution exacte des modèles linéaires pour les deux phases d'optimisation a été réalisée à l'aide du solveur CPLEX, en utilisant le langage de programmation mathématique AMPL. Il existe tout de même plusieurs solveurs pour des problèmes linéaires, toutefois notre choix s'est porté sur CPLEX pour la puissance du logiciel. Ce dernier s'est révélé efficace et performant dans la recherche de solutions optimales, malgré que nous n'ayons utilisé que les réglages de base du logiciel. En effet, une alternative pouvant améliorer davantage les performances du logiciel serait, par exemple, de développer dans un autre langage un algorithme de branchement plus performant, afin d'être en mesure d'augmenter encore plus la taille des instances traitées par la méthode exacte. Cependant, cela nous semblait non concluant sur notre travail, car une limite pour la résolution optimale se serait toujours présentée à un moment donné, dû à la nature NP-Difficile du problème.

Aussi, dans notre résolution exacte, la frontière entre les instances de petite taille et celles de grande taille découle tout simplement du travail réalisé dans le premier volet de la thèse, qui a dû limiter arbitrairement le temps d'exécution de l'approche exacte à 21,600 secondes. Toutes les instances de VNRs de plus de 60 VMs atteignant alors cette limite, nous avons délibérément borné les instances de petite taille à 70 VMs, et celle de grande taille à plus.

7.1.2.2 Approche de résolution approximative

Pour la résolution du problème du partitionnement des VNRs avec les instances de grande taille, plusieurs alternatives de résolution approximative, parmi celles passées en revue dans la littérature, se présentaient également. Toutefois, notre choix s'est naturellement d'abord porté sur la métaheuristique de TS, pourr son efficacité notoire dans le parcours de l'espace des solutions pour divers problèmes combinatoires complexes gérant de nombreuses contraintes, notamment grâce à ses mécanismes de mémoire flexibles permettant d'éviter les cycles et les optima locaux. Par la suite, dans l'objectif de pouvoir générer des solutions davantage de meilleure qualité, l'idée d'exploiter la technique d'hybridation de métaheuristiques de nature différente nous a paru judicieuse, à savoir combiner une métaheuristique de recherche locale à solution unique et une autre à base de population à solutions multiples, afin de tirer avantage des particularités de chacune des approches. En ce sens, notre première démarche s'était orientée sur l'adaptation de l'algorithme génétique qui, dans la littérature, est couramment hybridé avec une méthode de recherche locale, donnant lieu à l'algorithme mémétique qui est très connu. Cependant, l'algorithme génétique étant très gourmand en temps de calcul, nous avons opté finalement pour l'algorithme d'ACO. Ce dernier est certes rarement utilisé dans la littérature pour résoudre un tel problème, mais s'est avéré très efficace, autant en

temps de résolution et qu'en qualité des solutions fournies, notamment grâce au mécanisme centralisé basé sur l'algorithme de TS que nous y avons intégré.

Quant à la phase d'hébergement intraCloud des segments de VNRs de grande taille, une approche de résolution approximative n'est utilisée que dans le deuxième volet du travail, dans le but de confirmer l'efficacité de la stratégie de partitionnement proposée sur des instances de VNRs pouvant initialement contenir jusqu'à 500 VMs. À cette fin, nous avons simplement exécuté un algorithme de descente locale, afin de pouvoir rapidement réaliser les évaluations, les contributions étant, pour rappel, plus portées au niveau de la phase de partitionnement.

Aussi, les performances des différentes approches approximatives sont évaluées en comparant les solutions générées à celles obtenues avec l'approche exacte pour les instances de petite taille. Pour celles de grande taille, notre première démarche s'était dirigée à la définition de bornes (inférieure et supérieure), en procédant à des techniques de relaxation des contraintes d'intégrité du modèle. Cependant, les solutions obtenues déviaient considérablement, voire même devenaient irréalisables dans certains cas, par rapport à celles obtenues avec la méthode exacte ou les métaheuristiques. De ce fait, pour les instances de grande taille, comme communément ce qui est adopté par de nombreuses références dans la littérature, nous avons dû adapter à notre problème et implémenter des algorithmes connus, comme celui proposé par (Leivadeas *et al.*, 2013), et évaluer équitablement les solutions obtenues par les différents algorithmes avec la meilleure solution obtenue pour chacune des instances dans cette catégorie, parmi toutes les séries de simulation réalisées.

7.1.2.3 Environnement de simulation et méthodes d'évaluation du modèle

Pour l'évaluation de performance, aussi bien pour la stratégie de partitionnement que celle des approches de résolution, ne disposant pas d'un outil de simulation permettant de reproduire fidèlement le cadre de VNE proposé dans ce travail, nous avons dû concevoir notre propre simulateur de l'environnement multiCloud et intraCloud, ainsi que de l'arrivée intermittente des VNRs. Partant de là, nous avons généré les jeux de données et les instances nécessaires pour les différentes simulations, en nous basant sur certaines données utilisées dans la littérature. En particulier, pour l'évaluation de performance du modèle stratégique de partitionnement proposé, il n'était pas envisageable de comparer directement les résultats obtenus à ceux des approches de référence, les jeux de données et les modèles étant différents. De ce fait, nous avons implémenté les modèles cités dans notre simulateur, en les exécutant sur les mêmes données et instances que les nôtres, cette démarche nous offrant une base de comparaison équitable et adéquate.

Par ailleurs, il est naturel que tous les CPs intervenant dans le multiCloud n'utilisent pas, en réalité, nécessairement le même modèle pour l'hébergement des applications dans leur Cloud. Cependant, dans le cadre proposé, pour tous les modèles de partitionnement avec lesquels nous nous comparons et pour tous les CPs sélectionnés, nous avons appliqué la même approche de résolution pour l'intégration des segments de VNRs dans les infrastructures intraCloud. Cela nous semblait être plus équitable et judicieux, car permettant de mieux évaluer l'efficacité du modèle de partitionnement proposé, comparé à la littérature.

7.2 Analyse des résultats

L'analyse de l'ensemble des résultats obtenus nous permet d'apprécier de manière plus que satisfaisante la portée des différents travaux réalisés dans le cadre de cette thèse. Nous avons pu contribuer au développement d'un modèle d'optimisation de la sélection des CPs dans un environnement multiCloud, qui intègre l'ensemble des éléments techniques clés, des contraintes, des exigences et des intérêts liés aux différentes entités et acteurs en interactiton.

Ainsi qu'en attestent les résultats numériques présentés dans les chapitres précédents, le cadre de VNE multiCloud proposé a permis de nettement d'améliorer la performance et la QdS des demandes d'hébergement des applications dans les infrastructures Cloud. Le modèle stratégique de partitionnement des VNRs a pu faire augmenter en moyenne de 15.1 % le taux d'acceptation et diminuer le délai de communication d'environ 18.5 %, tout en réduisant également les violations de QdS d'environ 27.8 %. Aussi, les approches de résolution approximatives proposées résultent en un très bon compromis offert entre la qualité des solutions obtenues et le temps de résolution, en permettant, entre autres, de générer des solutions avec un écart de coût en moyenne à moins de 3.42 % de la solution exacte, et ce en un temps de calcul considérablement réduit pour un problème d'une telle complexité. Par ailleurs, tout en améliorant le rapport performance/coût d'environ 8 %, le cadre proposé permet d'offrir au SP un bon compromis entre performance optimisée et coûts d'approvisionnement réduits, à travers un modèle varié de sélection des ressources et services Cloud les mieux adaptés en fonction de ses intérêts et de ses objectifs. Quant aux CPs, les résultats obtenus sur les profits augmentés d'environ 67.2 %, pourraient les inciter à divulguer au VNP davantage d'informations sur les ressources et les services, et à être plus ouverts en matière d'interopérabilité, de manière à optimiser encore mieux l'utilisation de leurs ressources et d'augmenter ainsi le taux d'acceptation et les profits.

En conclusion, tout au long de la thèse, des choix à plusieurs niveaux de la démarche méthodologique ont dû être effectués dans le but d'atteindre les objectifs énoncés à la Section 1.3. Certains de ces choix ont été motivés par diverses raisons, d'autres ont été simplement la suite

logique de travaux préliminaires, qui n'ont pas nécessairement été présentés dans cette thèse. Dans le chapitre suivant, nous pourrons étudier les limites de la démarche méthodologique adoptée, lesquelles conduiront potentiellement à des travaux de recherche futurs.

CHAPITRE 8 CONCLUSION ET RECOMMANDATIONS

Ce dernier chapitre du manuscrit a pour but d'effectuer un récapitulatif des différents travaux réalisés dans le cadre de cette thèse. Nous débuterons par un rappel sommaire des principales contributions de la thèse. Par la suite, les limitations de nos travaux seront présentées, avant de clore le chapitre par des recommandations pouvant donner suite à de potentielles avenues de recherche future.

8.1 Sommaire des contributions de la thèse

Cette thèse visait principalement à concevoir un nouveau cadre hiérarchique d'optimisation de l'intégration des requêtes de réseaux virtuels (VNRs) dans un environnement multiCloud offrant des services sous le modèle IaaS. Le cadre développé inclut l'élaboration d'une première approche stratégique de partitionnement des VNRs parmi les multiples fournisseurs de Cloud (CPs) de l'environnement, et l'adoption d'une approche subséquente d'hébergement des différents segments de VNRs dans les infrastructures Cloud sélectionnées. En considérant la nature complexe du problème et le caractère dynamique de l'environnement multiCloud, l'objectif était de partitionner efficacement les VNRs selon les exigences et contraintes liées aux applications du fournisseur de service (SP), et d'optimiser conjointement la performance et la Qualité de Service (QdS) des machines virtuelles (VMs) et des liaisons virtuelles (VLs) à intégrer dans les différents réseaux Cloud, tout en minimisant les coûts induits d'approvisionnement de ressources et de services pour le SP.

Cette thèse a pu donner lieu à de nombreuses contributions, résumées ci-dessous, aussi bien dans la résolution globale de l'ensemble du processus du VNE dans un environnement multiCloud que dans le domaine des métaheuristiques :

1. **Modélisation mathématique d'applications complexes :** Nous avons pu proposer un cadre global d'intégration des VNRs dans un environnement multiCloud, qui permet de considérer et de modéliser des applications complexes encapsulées dans plusieurs VMs échangeant du trafic entre elles. Cela rajoute plus de complexité au problème comparé aux approches traditionnelles qui considèrent un modèle IaaS n'offrant uniquement que des services d'hébergement de VMs. Le cadre propose une modélisation des VNRs soumises par le SP, qui prend en compte tous les besoins en performance et les contraintes de QdS associés autant aux VMs à héberger qu'au trafic inter-VMs à router. La modélisation proposée intègre tous les types de ressources nécessaires à la

performance d'une VM, en pondérant en même temps l'importance de la quantité demandée pour chacune d'elles, de façon à optimiser le traitement d'une VM en fonction de l'application qu'elle encapsule. Les applications sensibles au délai sont également considérées, en limitant un délai maximal de communication entre certaines VMs.

2. **Modélisation mathématique de l'environnement multiCloud basée sur les politiques restrictives des CPs :** Une autre des contributions du travail réalisé repose sur la modélisation de l'environnement multiCloud et des informations collectées dans l'entrepôt de données, qui se conforment aux politiques restrictives des CPs. En effet, la plupart des modèles existants assument traditionnellement soit une connaissance complète et détaillée des réseaux sous-jacents, ou sinon utilisent néanmoins des informations confidentielles sur les CPs dans leur stratégie de partitionnement. Nous avons pu dans notre modélisation étudier le niveau de restriction quant à l'accès du fournisseur de réseau virtuel (VNP) aux informations sur l'environnement multiCloud, afin de concevoir une stratégie de partitionnement des VNRs basée sur les informations limitées et agrégées fournies par les CPs. Par ailleurs, dans le but de fournir une meilleure efficacité de la stratégie de partitionnement proposée, nous avons enrichi davantage la visibilité du VNP sur l'environnement multiCloud, en ajoutant dans la modélisation des données statistiques de performance garantie sur les réseaux de points de présence (PoPs) des CPs, qui ne sont pas considérées confidentielles.

3. **Conception d'un nouveau modèle de programmation mathématique pour le problème d'optimisation de partitionnement multiCloud des VNRs :** Des techniques de programmation linéaire en nombres entiers (ILP) ont été utilisées pour formaliser mathématiquement la stratégie de sélection des CPs proposée pour résoudre le problème du partitionnement des VNRs. Tout en considérant également la nature dynamique de ce dernier et les mises à jour périodiques des données publiées dans le référentiel d'informations, les différents paramètres exploités pour définir la fonction objectif et les contraintes linéaires du modèle, résultent de la modélisation préalablement réalisée des VNRs et de l'environnement multiCloud sous-jacent. Le modèle d'optimisation linéaire, dans un premier temps, maximise conjointement la performance et le niveau de QdS des VMs et des VLs, et ce aux localités géographiques de préférence du SP, tout en minimisant également le délai de communication inter-VMs et en évitant les violations de contraintes de délai maximal imposées sur les VLs. Par la suite, une extension du modèle mathématique est proposée, afin d'y intégrer les objectifs économiques du SP portant sur la minimisation des coûts d'approvisionnement de ressources et de services Cloud. La stratégie de sélection des CPs nous permet de fournir un modèle de partitionnement multiCloud des VNRs regroupant divers fac-

teurs de performance, de QdS et d'économie à optimiser pour le SP, ce qui offre à ce dernier le choix de sélectionner les CPs selon ses objectifs de préférence.

4. **Adoption d'un modèle de résolution multiobjectif du problème d'hébergement intraCloud des segments de VNRs :** Nous avons adopté une approche multiobjectif basée sur un modèle linéaire mixte (MILP), que nous avons adaptée au cadre de VNE multiCloud proposé, afin de pouvoir héberger convenablement dans les infrastructures Cloud sélectionnées, les segments de VNRs résultant du modèle d'optimisation de partitionnement proposé. Le modèle multiobjectif vise précisément à maximiser les profits des CPs, dont les revenus sont calculés en fonction de la durée de vie des VNRs acceptés dans leur Cloud, et les dépenses déterminées en fonction du coût d'utilisation des ressources serveur et réseau dans leur infrastructure, du coût de l'énergie, de l'impact écologique et du délai de routage du trafic dans leur réseau dorsal. L'implémentation d'une telle approche s'est avérée pertinente car permettant de refléter davantage la réalité du contexte multiCloud, mais également d'évaluer la stratégie de partitionnement appliquée sur plusieurs critères de performance, incluant le taux d'acceptation, le taux de partitionnement, le délai de communication, le temps d'exécution et les violations de contraintes de QdS, et sur les critères économiques relatifs au coût d'approvisionnement pour le SP et au profit pour les CPs.

5. **Développement de plusieurs approches approximatives basées sur les métaheuristiques, permettant de pallier la nature NP-Difficile du problème du partitionnement des VNRs :** Une autre contribution de la thèse constitue un apport autant dans l'élaboration des mécanismes avancés de résolution automatique du problème de partitionnement multiCloud des VNRs, mais également dans l'adaptation des métaheuristiques à cette classe de problème combinatoire. En effet, les différents modèles ILP proposés, ne permettant pas de traiter convenablement des instances de grande taille du problème avec une approche exacte, dû aux temps de calcul augmentant de manière exponentielle, nous avons élaboré deux approches approximatives de résolution basées sur les métaheuristiques. Dans un premier temps, une adaptation de la Recheche Taboue ou (TS) est proposée. Cette dernière nous a permis d'obtenir, en un temps réduit à une complexité polynomiale, des solutions (presque) optimales, grâce notamment aux mécanismes avancés de mémoire implémentés par la métaheuristique. Par la suite, nous avons développé une approche hybride basée sur une combinaison intégrative des métaheuristiques de colonies de fourmis (ACO) et de TS. Cette méthode, peu commune, basée sur des techniques d'hybridation de différentes métaheuristiques, donne l'avantage d'exploiter les capacités de chacune des méthodes combinées, afin d'explorer plus efficacement l'espace de recherche et d'obtenir des solutions davantage

de meilleure qualité. Les résultats obtenus avec l'approche hybride montrent des qualités des solutions meilleures que celles des différentes approches exécutées isolément, ce qui nous permet fournir une méthode de résolution offrant un excellent compromis entre qualité des solutions générées et temps d'exécution.

6. **Réduction considérable du temps d'exécution des approches heuristiques par la définition de fonctions de calcul avancées :** La dernière contribution de notre travail se rapporte à la méthode d'évaluation des configurations de solutions déterminées par les approches heuristiques proposées. En effet, au lieu de communément évaluer entièrement une solution à chaque modification de la configuration et à chaque itération, ce qui est très couteux en temps de calcul, nous avons défini des mécanismes de calcul de gains et de variations de coût, permettant d'accélérer considérablement le temps d'exécution dans la recherche de solutions. De ce fait, des fonctions de calcul de gains sont établies pour l'approche basée sur TS, afin d'évaluer uniquement la différence entre le coût d'une solution actuelle et celui d'une solution voisine. Pour l'approche hybride, dans les phases de construction progressive d'une configuration de solution par l'algorithme d'ACO, des fonctions de calcul de variations de coût sont définies pour uniquement évaluer la différence de coût générée par l'ajout d'un nouveau composant à la configuration partielle d'une solution actuelle. Cette technique, certes plus compliquée à implémenter, nous permet toutefois de réduire considérablement l'ordre de complexité du temps de calcul de $O(n^2)$ à $O(n)$.

8.2 Limitations des travaux réalisés

Malgré les nombreuses contributions énoncées ci-dessus, les travaux réalisés dans le cadre de cette thèse présentent tout de même certaines limitations :

— Le modèle d'optimisation proposé pour le partitionnement des VNRs est très dépendant des politiques de confidentialité des CPs, et de ce fait, de certaines informations publiées dans l'entrepôt de données qui pourraient ne plus être garanties, si par exemple le niveau de restriction s'élève davantage sur leur publication. En particulier, si les informations relatives à la disponibilité des ressources ou aux données de performance sur le réseau venaient à être dissimulées, notre modèle de sélection pourrait se ramener à un seul objectif possible pour le SP, à savoir la minimisation usuelle des coûts d'approvisionnement, ce qui ne lui donnerait pas l'opportunité de pouvoir sélectionner les CPs selon la performance et la QdS de ses applications.

— Une autre limitation du travail réside dans la gestion de la dynamicité du traitement des requêtes. Dans notre cadre de simulation, nous traitons certes les requêtes à leur

arrivée, et nous libérons les ressources allouées dès leur expiration pour en prendre compte dans les traitements futurs. Cependant, nous n'avons pas considéré les cas où elles arrivent en même temps pour envisager de les traiter simultanément. Les requêtes sont étudiées une à la fois, sans dépendance ou concurrence entre elles. Une telle considération de dynamicité dans notre travail nécessiterait la mise en œuvre de mécanismes d'optimisation dynamique, afin de pouvoir re-assigner ou re-localiser les requêtes existantes ou en cours de traitement, pour assurer la continuité de service dans la satisfaction des besoins en ressources des applications concurrentes. De même, notre cadre de VNE s'insère dans un contexte où les besoins en ressources des applications soumises sont fixes et ne varient pas dans le temps. Afin de mieux refléter la réalité dans un tel environnement, cette considération dynamique pourrait également être intégrée dans le cadre proposé.

— Une autre des limitations peut être observée au niveau de la modélisation des VNRs, où seul le trafic entre les VMs est pris en compte dans nos modèles et non celui, non négligeable tout de même, entre les VMs et les utilisateurs finaux des services déployés par le SP. Certes, cette omission n'engendre pas une perte de généralité autour du cadre de VNE proposé, car ce type de trafic peut sans difficulté être intégré dans nos modèles. Toutefois, il serait intéressant de le considérer afin d'enrichir davantage l'applicabilité du cadre de VNE conçu dans le contexte du modèle IaaS du Cloud Computing.

— Bien que nous ayons considéré, dans le modèle d'optimisation du partitionnement, des prix unitaires de ressources variant par période de temps, pour chacune de ces dernières, le prix est connu à l'avance et demeure fixe durant la résolution du problème et pendant la durée de vie des VNRs. Il serait plus intéressant de pouvoir intégrer dans le modèle des mécanismes de tarification plus dynamique, pouvant être variante selon la durabilité des requêtes.

— Une autre limitation repose au niveau de la résolution du problème de l'hébergement intraCloud des applications après la phase de sélection des CPs. En effet, notre résolution s'effectuant à l'échelle de la sélection des DCs, afin de pouvoir procurer des résultats encore plus précis à ce niveau de la hiérarchie du processus global, une approche de résolution de l'hébergement devrait pouvoir se poursuivre également au niveau de la sélection des serveurs internes aux DCs. Cela permettrait de mieux évaluer les métriques de performance considérées par le CP dans son modèle d'optimisation intraCloud.

— Également, l'efficacité du modèle de partitionnement proposé est évaluée en exécutant la même approche de résolution pour la phase d'hébergement intraCloud des applications chez tous les CPs sélectionnés. Cette démarche a été motivée uniquement par notre objectif d'évaluer la stratégie de sélection de CPs élaborée plus équitablement et dans un contexte plus simplifiée de simulation, afin de mieux en apprécier son efficacité. Dans la réalité, chacun des CPs étant libre de procéder à une approche d'optimisation de son choix pour héberger les applications qui lui sont assignées dans son réseau, il serait pertinent aussi d'envisager des cas où les modèles exécutés pour chaque CP de l'environnement soient différents, afin d'étudier l'impact d'une tel scénario sur les performances du modèle stratégique de partitionnement proposé.

— Une autre limitation du travail est associée à la non définition de bornes pour évaluer les performances des approches heuristiques proposées sur les instances de grande taille du problème. En effet, pour les instances de petite taille que la méthode exacte a pu traiter, le souci ne se pose pas car la proximité des solutions générées par les approches heuristiques a pu être évaluée convenablement par rapport à la solution optimale. Cependant, pour des instances de grande taille, il nous a fallu implémenter d'autres algorithmes approximatifs de la littérature et évaluer les performances de nos approches heuristiques par rapport à ces derniers, en nous basant sur la meilleure solution trouvée parmi toutes les séries de simulation. Toutefois, cette démarche ne permettant pas de garantir une proximité appréciable par rapport à la solution optimale inconnue, il serait alors plus pertinent de pouvoir mettre en œuvre des mécanismes de détermination de bornes pour les approches approximatives proposées.

8.3 Travaux futurs

Nous ne pourrions clore cette thèse sans proposer des avenues de recherche intéressantes pouvant faire l'objet de travaux futurs. Certaines des propositions permettront, sans nul doute, de surmonter les limites observées précédemment :

— Une première voie de recherche pourrait s'orienter vers le développement d'une plateforme standardisée et interopérable entre les CPs intervenant dans l'environnement multiCloud. La plateforme serait en mesure de recouvrir le référentiel d'informations utilisé par le VNP, dont certaines données sensibles pour le CP pourraient être encodées par le biais de mécanismes de cryptage multipartite. Cela pourrait persuader les CPs de maintenir, voire même renforcer davantage, le niveau de visibilité du VNP considéré dans ce travail. En effet, nos résultats étant concluant que les CPs ont tout un intérêt économique à dévoiler davantage d'informations sur le réseau, une plate-

forme semblable pourrait les inciter dans cette voie, tout en s'assurant de préserver leur confidentialité.

— Une piste importante à analyser serait de proposer un modèle d'optimisation plus dynamique et adaptatif. Ce dernier permettrait déjà de traiter plusieurs requêtes concurrentes simultanément. Ensuite, un tel modèle d'optimisation pourrait intégrer des mécanismes de re-allocation de ressources, qui seraient en mesure de s'adapter aussi à des demandes en performance incertaine et évoluant dans le temps pour les applications soumises par le SP.

— Des aspects dynamiques pourraient également s'appliquer au niveau du modèle de tarification utilisé, afin de mieux refléter la réalité. En effet, en plus de considérer une charge incertaine des applications à traiter, prendre en compte une incertitude sur les tarifs imposés par les CPs par rapport aux ressources et services Cloud serait une piste tout aussi pertinente à explorer. Une dynamicité semblable dans le modèle de tarification pourrait même également s'inscrire dans un cadre de VNE, où chacune des différentes parties en interaction dans l'environnement optimise de manière conflictuelle et compétitive leurs fonctions d'utilité respectives.

— Aussi, l'exploitation de techniques d'apprentissage automatique constituerait une avenue de recherche, certes s'annonçant complexe, mais intéressante pour ce problème d'optimisation. Cela se traduirait par la mise en œuvre d'approches basées sur l'apprentissage de données et d'expériences collectées, qui permettrait de fournir un système autonome et optimisé de prise de décisions de manière adaptative et prédictive, face à une nature dynamique et incertaine des entités de l'environnement multiCloud.

— Un travail pertinent relatif à l'évaluation des performances des approches heuristiques proposées serait d'élaborer un mécanisme de calcul de bornes de bonne qualité pour les instances de grande taille du problème. Le mécanisme pourrait, par exemple, utiliser une équation mathématique dont le calcul de la valeur pour une borne se ferait en un temps raisonnable, ou alors utiliser des techniques de relaxation plus avancées.

— Pour terminer, il serait intéressant d'envisager la possibilité d'intégrer l'ensemble de nos travaux dans une plateforme réelle ou dans un simulateur conforme multiCloud, afin de pouvoir considérer toutes les propriétés d'un environnement semblable dans les modèles d'optimisation proposés et d'en valider plus convenablement la faisabilité.

RÉFÉRENCES

ADDYA, S. K., TURUK, A. K., SAHOO, B., SARKAR, M. et BISWASH, S. K. (2017). Simulated annealing based vm placement strategy to maximize the profit for cloud service providers. *Engineering Science and Technology, an International Journal*, 20, 1249–1259.

AMAZON (2017). Amazon EC2 Instance Types. [En ligne]. Disponible : http ://aws.amazon. com/ec2/instance- types/. Visité le : 13/12/2017.

AMOKRANE, A., LANGAR, R., ZHANI, M. F., BOUTABA, R. et PUJOLLE, G. (2015). Greenslater : On satisfying green slas in distributed clouds. *IEEE Transactions on Network and Service Management*, 12.

AMOKRANE, A., ZHANI, M. F., LANGAR, R., BOUTABA, R. et PUJOLLE, G. (2013). Greenhead : Virtual Data Center Embedding across Distributed Infrastructures. *IEEE Transactions on Cloud Computing*, 1.

ARAL, A. et OVATMAN, T. (2016). Network-aware embedding of virtual machine clusters onto federated cloud infrastructure. *The Journal of Systems and Software*, 120, 89–104.

ARAÚJO, S. M. A., DE SOUZA, F. S. H. et MATEUS, G. R. (2018). Virtual network embedding in multi-domain environments with energy efficiency concepts. *IEEE International Conference on Information Networking (ICOIN)*.

ARDAGNA, D., CIAVOTTA, M. et PASSACANTANDO, M. (2017). Generalized Nash Equilibria for the Service Provisioning Problem in Multi-Cloud Systems. *IEEE Transactions on Services Computing*, 10, 381–395.

ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R., KONWINSKI, A., LEE, G., PATTERSON, D., RABKIN, A., STOICA, I. et ZAHARIA, M. (2010). A View of Cloud Computing. *Communications of the ACM*, 53, 50–58.

AYOUBI, S., ASSI, C., SHABAN, K. et NARAYANAN, L. (2015). MINTED : Multicast VIrtual NeTwork Embedding in Cloud Data Centers With Delay Constraints. *IEEE Transactions on Communications*, 63, 1291–1305.

AYOUBI, S., ZHANG, Y. et ASSI, C. (2016). A reliable embedding framework for elastic virtualized services in the cloud. *IEEE Trans. on Netw. and Serv. Manag.*, 13, 489–503.

BELBEKKOUCHE, A., HASAN, M. M. et KARMOUCH, A. (2012). Resource Discovery and Allocation in Network Resource Virtualization. *IEEE Communications Surveys and Tutorials*, 14, 1114–1128.

BELLUR, U., MALANI, A. et NARENDRA, N. C. (2014). Cost Optimization in Multi-site Multi-cloud Environments with Multiple Pricing Schemes. *2014 IEEE 7th International Conference on Cloud Computing (CLOUD)*. 689–696.

BHAMARE, D., JAIN, R., SAMAKA, M., VASZKUN, G. et ERBAD, A. (2015). Multi-cloud distribution of virtual functions and dynamic service deployment : Open ADN perspective. *Cloud Engineering (IC2E), 2015 IEEE International Conference on*. IEEE, 299–304.

BLUM, C., PUCHINGER, J., RAIDL, G. R. et ROLI, A. (2011). Hybrid metaheuristics in combinatorial optimization : A survey. *Applied Soft Computing*, 11, 4135–4151.

BLUM, C. et ROLI, A. (2003). Metaheuristics in Combinatorial Optimization : Overview and Conceptual Comparison. *ACM Comput. Surv.*, 35, 268–308.

BOUSSAïD, I., LEPAGNOT, J. et SIARRY, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237, 82–117.

CAO, X., POPESCU, I., CHEN, G., GUO, H., YOSHIKANE, N., TSURITANI, T., WU, J. et MORITA, I. (2017). Optimal and dynamic virtual datacenter provisioning over metro-embedded datacenters with holistic SDN orchestration. *Optical Switching and Networking*, 24.

CARDELLINI, V., VALERIO, V. D. et PRESTI, F. L. (2018). Game-Theoretic Resource Pricing and Provisioning Strategies in Cloud Systems. *IEEE Transactions on Services Computing*, 1–1.

CHAISIRI, S., LEE, B.-S. et NIYATO, D. (2012). Optimization of resource provisioning cost in cloud computing. *IEEE Transactions on Services Computing*, 5, 164–177.

CHANG, X., MI, X. et MUPPALA, J. (2013). Performance evaluation of artificial intelligence algorithms for virtual network embedding. *Engineering Applications of Artificial Intelligence*, 26, 2540–2550.

CHOWDHURY, M., RAHMAN, M. R. et BOUTABA, R. (2012). Vineyard : Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking*, 20, 206–219.

COULLON, H., LOUET, G. L. et MENAUD, J.-M. (2017). Virtual Machine Placement for Hybrid Cloud Using Constraint Programming. *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*. 326–333.

CPLEX (2018). IBM ILOG CPLEX Optimizer. [Online]. Available : https ://www.ibm.com/products/ilog-cplex-optimization-studio. Visité le : 03/02/2018.

DIALLO, M., QUINTERO, A. et PIERRE, S. (2018a). A QoS-based splitting strategy for a resource embedding across multiple cloud providers. *IEEE Transactions on Services Computing*. (forthcoming).

DIALLO, M., QUINTERO, A. et PIERRE, S. (2018b). A Tabu search approach for a virtual networks splitting strategy across multiple cloud providers. *International Journal of Metaheuristics*. (forthcoming).

DIALLO, M., QUINTERO, A. et PIERRE, S. (2018c). Two Efficient QoS-based Approaches for a Resource Splitting Strategy Across Multiple Cloud Providers. *International Conference on Utility and Cloud Computing (UCC), 2018 ACM/IEEE 11th Conference on*. (forthcoming).

DIETRICH, D., RIZK, A. et PAPADIMITRIOU, P. (2015). Multi-provider virtual network embedding with limited information disclosure. *IEEE Transactions on Network and Service Management*, 12, 188–201.

DORIGO, M. et STÜTZLE, T. (2010). *Ant colony optimization : overview and recent advances*, Handbook of Metaheuristics, vol. 146. 227–263.

D'ORO, S., GALLUCCIO, L., MERTIKOPOULOS, P., MORABITO, G. et PALAZZO, S. (2017). Auction-based resource allocation in openflow multi-tenant networks. *Computer Networks*, 115, 29–41.

DOVERSPIKE, R. D., RAMAKRISHNAN, K. et CHASE, C. (2010). Structural overview of ISP networks. *C. Kalmanek, S. Misra, R. Yang (Editors.), Guide to Reliable Internet Services and Applications*, Springer, London. 19–93.

FISCHER, A., BOTERO, J. F., BECK, M. T., DE MEER, H. et HESSELBACH, X. (2013). Virtual Network Embedding : A Survey. *IEEE Communications Surveys and Tutorials*, 15, 1888–1906.

GHALEB, A. M., KHALIFA, T., AYOUBI, S., SHABAN, K. B. et ASSI, C. (2016). Surviving Multiple Failures in Multicast Virtual Networks With Virtual Machines Migration. *IEEE Transactions on Network and Service Management*, 13, 899–912.

GLOVER, F. (1989). Tabu Search—Part I. *ORSA J. Computing*, 1, 190–206.

GLOVER, F. (1990). Tabu Search—Part II. *ORSA J. Computing*, 2, 4–32.

GONG, S., CHEN, J., YIN, X. et ZHU, Q. (2016). Survivable virtual network embedding across multiple domains. *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. 2391–2396.

GROZEV, N. et BUYYA, R. (2012). Inter-Cloud architectures and application brokering : taxonomy and survey. *Software - Practice and Experience*, 44, 369–390.

GUAN, X., WAN, X., CHOI, B.-Y. et SONG, S. (2015). Ant Colony Optimization Based Energy Efficient Virtual Network Embedding. *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*. 273–278.

HE, M., ZHUANG, L., TIAN, S., WANG, G. et ZHANG, K. (2018). Multi-objective virtual network embedding algorithm based on Q-learning and curiosity-driven. *EURASIP Journal on Wireless Communications and Networking*, 150.

HEILIG, L., LALLA-RUIZ, E. et VOß, S. (2016). A cloud brokerage approach for solving the resource management problem in multi-cloud environments. *Computers and Industrial Engineering*, 95, 16–26.

HESSELBACH, X., AMAZONAS, J. R., VILLANUEVA, S. et BOTERO, J. F. (2016). Coordinated node and link mapping VNE using a new paths algebra strategy. *Journal of Network and Computer Applications*, 69, 14–26.

HOUIDI, I., LOUATI, W., AMEUR, W. B. et ZEGHLACHE, D. (2011). Virtual network provisioning across multiple substrate networks. *Computer Networks*, 55, 1011–1023.

HOUIDI, I., LOUATI, W. et ZEGHLACHE, D. (2015). Exact multi-objective virtual network embedding in cloud environments. *The Computer Journal*, 58, 403–415.

IBM (2014). Cloud computing defined : Characteristics and service levels. [En ligne]. Disponible : https ://www.ibm.com/blogs/cloud-computing/2014/01/31/cloud-computing-defined-characteristics-service-levels/. Visité le : 19/11/2014.

INFÜHR, J. et RAIDL, G. (2016). A memetic algorithm for the virtual network mapping problem. *Journal of Heuristics*, 22, 475–505.

JUSTAFORT, V. D., BEAUBRUN, R. et PIERRE, S. (2015). On the carbon footprint optimization in an intercloud environment. *IEEE Transactions on Cloud Computing*, 6, 829–842.

JUSTAFORT, V. D., BEAUBRUN, R. et PIERRE, S. (2016). A Hybrid Approach for Optimizing Carbon Footprint in InterCloud Environment. *IEEE Transactions on Services Computing*, 1–1.

JUSTAFORT V. D., BEAUBRUN R. ET PIERRE, S. (2015). An Iterated Local Search Approach for Carbon Footprint Optimisation in an Intercloud Environment. *International Journal of Metaheuristics*, 4, 159–184.

KHAN, M. M. A., SHAHRIAR, N., AHMED, R. et BOUTABA, R. (2016). Multi-Path Link Embedding for Survivability in Virtual Networks. *IEEE Transactions on Network and Service Management*, 13, 253–266.

KHOSHKHOLGHI, M. A., DERAHMAN, M. N., ABDULLAH, A., SUBRAMANIAM, S. et OTHMAN, M. (2017). Energy-Efficient Algorithms for Dynamic Virtual Machine Consolidation in Cloud Data Centers. *IEEE Access*, 5, 10709–10722.

LARUMBE, F. et SANSÒ, B. (2012). Cloptimus : A multi-objective cloud data center and software component location framework. *2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET)*. 23–28.

LARUMBE, F. et SANSO, B. (2013). A tabu search algorithm for the location of data centers and software components in green cloud computing networks. *IEEE Transactions on Cloud Computing*, 1, 22–35.

LEIVADEAS, A., PAPAGIANNI, C. et PAPAVASSILIOU, S. (2013). Efficient resource mapping framework over networked clouds via iterated local search-based request partitioning. *IEEE Transactions on Parallel and Distributed Systems*, 24, 1077–1086.

LI, S., SAIDI, M. Y. et CHEN, K. (2016). A Cloud-oriented Algorithm for Virtual Network Embedding Over Multi-Domain. *Local Computer Networks Workshops IEEE 41st Conference*.

LV, B., WANG, Z., HUANG, T., CHEN, J. et LIU, Y. (2010). Virtual resource organization and virtual network embedding across multiple domains. *2010 IEEE International Conference on Multimedia Information Networking and Security (MINES)*. 725–728.

MANO, T., INOUE, T., IKARASHI, D., HAMADA, K., MIZUTANI, K. et AKASHI, O. (2016). Efficient virtual network optimization across multiple domains without revealing private information. *IEEE Transactions on Network and Service Management*, 13, 477–488.

MANVI, S. S. et SHYAM, G. K. (2014). Resource management for Infrastructure as a Service (IaaS) in cloud computing : A survey. *Journal of Network and Computer Applications*, 41, 424–440.

MECHTRI, M., HADJI, M. et ZEGHLACHE, D. (2017). Exact and Heuristic Resource Mapping Algorithms for Distributed and Hybrid Clouds. *IEEE Transactions on Cloud Computing*, 5, 681–696.

MELL, P. et GRANCE, T. (2009). The NIST definition of cloud computing. *National Institute of Standards and Technology*, 53, 50.

MELO, M., SARGENTO, S., KILLAT, U., TIMM-GIEL, A. et CARAPINHA, J. (2013). Optimal virtual network embedding : Node-link formulation. *IEEE Transactions on Network and Service Management*, 10, 356–368.

MELO, M., SARGENTO, S., KILLAT, U., TIMM-GIEL, A. et CARAPINHA, J. (2015). Optimal virtual network embedding : Energy aware formulation. *Computer Networks*, 91, 184–195.

MIJUMBI, R., GORRICHO, J.-L., SERRAT, J., CLAEYS, M., TURCK, F. D. et LATRÉ, S. (2014). Design and evaluation of learning algorithms for dynamic resource management

in virtual networks. *2014 IEEE Network Operations and Management Symposium (NOMS)*. 1–9.

MURUGESAN, S. et BOJANOVA, I. (2016). *Encyclopedia of Cloud Computing*, vol. 744. Wiley-IEEE Press.

NIA, N. H., ADABI, S. et NATEGH, M. N. (2017). A coordinated heuristic approach for virtual network embedding in cloud infrastructure. *KSII Transactions on Internet and Information Systems*, 11, 2346–2361.

PAPAGIANNI, C., LEIVADEAS, A., PAPAVASSILIOU, S., MAGLARIS, V., CERVELLO-PASTOR, C. et MONJE, A. (2013). On the optimal allocation of virtual resources in cloud computing networks. *IEEE Transactions on Computers*, 62, 1060–1071.

PATHAK, I. et VIDYARTHI, D. P. (2017). A model for virtual network embedding across multiple infrastructure providers using genetic algorithm. *Science China Information Sciences*, 60.

PYOUNG, C. K. et BAEK, S. J. (2018). Joint load balancing and energy saving algorithm for virtual network embedding in infrastructure providers. *Computer Communications*, 121, 1–18.

RABAH, S., EL BARACHI, M., KARA, N., DSSOULI, R. et PAQUET, J. (2015). A service oriented broker-based approach for dynamic resource discovery in virtual networks. *Journal of Cloud Computing*, 4, 3.

RAFAEL, M.-V., RUBÉN, S. M. et IGNACIO, M. L. (2012). IaaS Cloud Architecture : From Virtualized Datacenters to Federated Cloud Infrastructures. *IEEE Computer Society*, 45, 65–72.

RAHMAN, M. R. et BOUTABA, R. (2013). SVNE : Survivable virtual network embedding algorithms for network virtualization. *IEEE Transactions on Network and Service Management*, 10, 105–118.

RAN, Y., YANG, B., CAI, W., XI, H. et YANG, J. (2016). Cost-Efficient Provisioning Strategy for Multiple Services in Distributed Clouds. *IEEE International Conference on Cloud Computing Research and Innovations*.

SAMUEL, F., CHOWDHURY, M. et BOUTABA, R. (2013). Polyvine : policy-based virtual network embedding across multiple domains. *Journal of Internet Services and Applications*, 4, 1–23.

SANCHIS, L. (1989). Multiple-Way Network Partitioning. *IEEE Trans. Computers*, 38, 62–81.

SHAHRIAR, N., AHMED, R., CHOWDHURY, S. R., KHAN, A., BOUTABA, R. et MITRA, J. (2017). Generalized Recovery from Node Failure in Virtual Network Embedding. *IEEE Transactions on Network and Service Management*, PP, 1–1.

STATISTA (2018). The statistics portal. [En ligne]. Disponible : https ://www.statista.com/statistics/258718/market-growth-forecast-of-public-it-cloud-services-worldwide/. Visité le : 04/03/2018.

TAO, L., ZHAO, C., THULASIRAMAN, K. et SWAMY, M. (1992). Simulated Annealing and Tabu Search Algorithms for Mulitway Graph Partitioning. *Journal of Circuits, Systems and Computers*, 02, 159–185.

WANG, C., SU, Y., ZHOU, L., PENG, S., YUAN, Y. et HUANG, H. (2017a). A Virtual Network Embedding Algorithm Based on Hybrid Particle Swarm Optimization. *International Conference on Smart Computing and Communication*. vol. 10135.

WANG, J.-B., CHEN, W.-N., CONG, H., ZHAN, Z.-H. et ZHANG, J. (2017b). An Ant Colony System Based Virtual Network Embedding Algorithm. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 1805–1810.

WANG, W., NIU, D., LI, B. et LIANG, B. (2013). Dynamic Cloud Resource Reservation via Cloud Brokerage. *2013 IEEE 33rd International Conference on Distributed Computing Systems*. 400–409.

YAO, H., CHEN, X., LI, M., ZHANG, P. et WANG, L. (2018). A novel reinforcement learning algorithm for virtual network embedding. *Neurocomputing*, 284, 1–9.

ZENG, L., VEERAVALLI, B. et WEI, Q. (2014). Space4time : Optimization latency-sensitive content service in cloud. *Journal of Network and Computer Applications*, 41, 358–368.

ZHANG, J., HUANG, H. et WANG, X. (2016a). Resource provision algorithms in cloud computing : A survey. *Journal of Network and Computer Applications*, 64, 23–42.

ZHANG, P., YAO, H., FANG, C. et LIU, Y. (2016b). Multi-objective enhanced particle swarm optimization in virtual network embedding. *EURASIP Journal on Wireless Communications and Networking*, 167.

ZHANG, P., YAO, H., LI, M. et LIU, Y. (2017). Virtual network embedding based on modified genetic algorithm. *Peer-to-Peer Networking and Applications, Springer*, 1–12.

ZHANG, S., QIAN, Z., WU, J., LU, S. et EPSTEIN, L. (2014). Virtual network embedding with opportunistic resource sharing. *IEEE Transactions on Parallel and Distributed Systems*, 25, 816–827.

ZHU, F. et WANG, H. (2016). A modified ACO algorithm for virtual network embedding based on graph decomposition. *Computer Communications*, 80, 1–15.

ZONG, Y., OU, Y., HAMMAD, A., KONDEPU, K., NEJABATI, R., SIMEONIDOU, D., LIU, Y. et GUO, L. (2018). Location-Aware Energy Efficient Virtual Network Embedding in Software-Defined Optical Data Center Networks. *J. OPT. COMMUN. NETW.*, 10.

ZOU, T., LE BRAS, R., SALLES, M. V., DEMERS, A. et GEHRKE, J. (2015). ClouDiA : a deployment advisor for public clouds. *The VLDB Journal*, 24, 633–653.