Le voyageur de commerce et ses variations : un tour d'horizon de ses résolution

Jérôme Monnot, Sophie Toulouse

Optimisation combinatoire: Problèmes paradigmatiques et problématiques nouvelles

Vangelis Th. PASCHOS

13 décembre 2006

Table des matières

Chapitre 1

Le voyageur de commerce et ses variations : un tour d'horizon de ses résolutions

1.1. Introduction

Toute personne qui se confronte un jour à la recherche opérationnelle croise nécessairement sur son chemin le problème du voyageur de commerce (noté usuellement TSP). Ce qui donne ce caractère incontournable au TSP, c'est à la fois sa facilité d'appréhension et sa proximité conjointe avec des problèmes simples et complexes. *Un problème parlant* : on imagine aisément le représentant de commerce tracer son parcours sur une carte. *Un problème de référence en optimisation combinatoire* : résoudre l'énigme **P=NP**, c'est trouver un algorithme polynomial pour TSP ou, au contraire, prouver que tout algorithme qui le résout de façon exacte est de complexité exponentielle. *Un problème pédagogique* : sa ressemblance aux problèmes d'affectation ou de 2-couplage permet de saisir la nuance qui sépare ce qui est facile de ce qui est non soluble, de cerner la nature et la difficulté intrinsèque de la combinatoire. Pour ces raisons sûrement, les chercheurs se sont, aux heures d'envol de la recherche opérationnelle, passionnés pour le TSP et de fait, son étude a permis l'émergence de nombreuses méthodes et méthodologies de résolution. Ainsi, parler du TSP, c'est non seulement expliciter ce qu'est l'optimisation combinatoire, ses difficultés et ses enjeux, mais c'est aussi le moyen de revisiter les méthodes de résolution présentées au cours des autres chapitres de cet ouvrage. C'est enfin montrer l'ingéniosité des chercheurs, la multiplicité des modèles mis au point et les réalités que ceux-ci incarnent.

Chapitre rédigé par Jérôme MONNOT et Sophie TOULOUSE.

Le problème d'affectation consiste à déterminer, dans un graphe, un ensemble de cycles deux à deux disjoints qui soit de poids minimal ; ce problème est polynomial. Entre une affectation et un tour du voyageur de commerce, la seule différence est une contrainte supplémentaire imposée par ce dernier : ne pas disposer de sous-cycle dans la solution. Lorsque l'on modélise ces problèmes sous forme de programme linéaire, cette différence à l'apparence anodine se traduit par l'ajout d'un nombre exponentiel de contraintes ; mais aussi, par la contrainte d'intégrité : $x_{ij} \in \{0, 1\}$, où la variable $x_{ij} \in \{0, 1\}$ traduit le choix d'intégrer ou non à la solution l'arête $[i, j]$. En réalité, cette contrainte est également vraie pour l'affectation (on choisit également de prendre ou non une arête), seulement on peut s'en affranchir car il est prouvé que les solutions optimales sont entières, les sommets du polyèdre dessiné par les contraintes du problème d'affectation étant des points aux coordonnées entières. Ainsi, cette contrainte de non-cyclage transforme le problème *'arrangeant'* d'affectation en le problème (le plus visité avec SAT peut-être) intraitable qu'est TSP.

La difficulté de résolution du TSP a d'autant plus porté sur lui l'attention des chercheurs. Aussi a-t-il été étudié et retourné sous tous les angles : théorie des graphes, programmation linéaire, programmation dynamique, optima locaux, etc. Les premières formalisations de la recherche exhaustive par la stratégie d'évaluation et séparation seraient même nées de recherches sur le voyageur de commerce. Ce chapitre retrace, certainement pas de façon exhaustive, l'histoire conjointe de la recherche opérationnelle et du voyageur de commerce. Après avoir présenté le problème, nous proposons des algorithmes, exacts puis approchés, pour différentes versions du problème : minimisation ou maximisation, instances métriques, distances binaires, etc. Certains de ces algorithmes mettent en œuvre des modèles généraux de résolution tels que la stratégie par séparation et évaluation, la programmation dynamique ou la recherche locale. Certains encore utilisent des heuristiques, qui ne sont autres que des idées de bon sens quant à la constitution d'une solution pour le problème étudié : nous pensons par exemple aux heuristiques du regret et du plus proche voisin. D'autres enfin, exploitant la relative facilité de sous-problèmes du TSP, partent d'une solution de ces sous-problèmes et construisent à partir de celle-ci un cycle hamiltonien ; c'est le parti pris par l'algorithme de Christofides avec l'arbre couvrant, mais de nombreux résultats sont également obtenus par le biais d'un 2-couplage optimal.

1.2. Propriétés élémentaires et différents sous-problèmes

1.2.1. *Propriétés élémentaires*

Un *cycle hamiltonien* T sur un graphe complet K_n à n sommets est un ensemble de n arêtes, noté $E(T)$. Cet ensemble d'arêtes doit vérifier que le graphe partiel $G' = (V(K_n), E(T))$ de K_n qu'il engendre consiste en un cycle unique qui couvre tous les sommets.

Il existe d'autres manières, plus intuitives peut-être, de caractériser les cycles hamiltoniens de K_n. La première est graphique et fait appel à la notion de parcours d'un cycle : un *parcours* d'un tour T de K_n est une séquence de sommets $(v_{i_1}, \ldots, v_{i_n})$ indiquant les sommets rencontrés lors de la description du cycle T. Précisons qu'un tour T connaît plusieurs parcours possibles, en fonction du point de départ du parcours, c'est-à-dire du premier sommet v_{i_1} visité, ainsi que du sens de ce parcours ; à l'inverse, à chaque parcours correspond un unique cycle hamiltonien.

La deuxième alternative de définition d'un cycle hamiltonien est fonctionnelle et fait appel aux permutations f sur $V(K_n)$. Nous rappelons qu'une permutation sur un ensemble est une application bijective de cet ensemble dans lui-même. Un cycle hamiltonien est alors une *permutation acyclique* f sur $V(K_n)$, soit une permutation qui vérifie la propriété :

$$\forall v \in V(K_n), \forall k \in \{1, \ldots, n-1\},\ f^k(v) \neq v \qquad [1.1]$$

où $f^k(v) = f \circ f^{k-1}(v) = f[f^{k-1}(v)]$. Remarquons que nous avons nécessairement $f^n(v) = v$ pour tout sommet $v \in V(K_n)$. Nous laissons le soin au lecteur de prouver que ces formulations sont équivalentes.

1.2.2. *Différents sous-problèmes*

Lorsque l'on parle du voyageur de commerce, on pense souvent à MIN TSP ; pourtant, nombreuses sont ses versions qui ont largement inspiré la littérature : lorsque la fonction distance vérifie l'inégalité triangulaire, [CHR 76] ou ses versions relaxées et restreintes, [BEN 99, BÖC 00], lorsque l'instance est géométrique, [ARO 98, BEN 92] ou encore, lorsque l'on cherche à maximiser l'objectif, cf. [FIS 79, HAS 00]. La fonction distance d vérifie l'*inégalité triangulaire* sur le graphe complet K_n lorsque l'on a la propriété :

$$\forall v_1, v_2, v_3 \in V(K_n),\ d(v_1, v_3) \leq d(v_1, v_2) + d(v_2, v_3) \qquad [1.2]$$

En d'autres termes, la plus courte distance entre deux villes est la 'ligne droite'. On parle alors d'*instance métrique* et la restriction du voyageur de commerce à ces instances est notée MIN METRIC TSP. Les instances géométriques vérifient bien entendu cette propriété, avec l'exigence supplémentaire que les sommets soient des coordonnées dans le plan (ou dans un espace de plus grande dimension) et que la distance entre deux points $x = (x_1, x_2)$ et $y = (y_1, y_2)$ soit issue d'une norme, par exemple $d_2(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ ou $d_1(x, y) = |x_1 - y_1| + |x_2 - y_2|$. On parle

dans le premier cas de distance *euclidienne*, dans le second de distance *rectiligne*. Finalement, lorsque l'objectif est de trouver un cycle hamiltonien dont la valeur est non plus la plus petite, mais la plus grande possible, on parle du problème MAX TSP, qui connaît ses propres applications industrielles, [BAR 02, GAR 85].

Les versions MIN TSP, MIN METRIC TSP et MAX TSP sont liées par les relations suivantes : pour toute instance $I = (K_n, d)$ de MIN TSP, on peut construire une instance $I' = (K_n, d')$ de MIN METRIC TSP ou de MAX TSP qui vérifie, pour tout cycle hamiltonien T de K_n, que la valeur de ce tour sur les deux instances sont égales à une transformation affine près. Plus que cela, de l'instance initiale I à l'instance transformée I', on aura préservé l'ordre relatif des solutions, vis-à-vis du critère d'optimisation considéré (minimiser ou maximiser). Concrètement, si nous prenons l'exemple du passage de MIN METRIC TSP à MAX TSP, on aura pour toute paire de cycles hamiltoniens T_1 et T_2 : $d(T_1) \leqslant d(T_2)$ sur I (c'est-à-dire, T_1 meilleure que T_2 sur I au sens de MIN TSP) *si et seulement si* $d(T_1) \geqslant d(T_2)$ sur I' (c'est-à-dire, T_1 meilleure que T_2 sur I au sens de MAX TSP).

Par la suite, on notera $d_{max} = \max_{e \in E(K_n)} d(e), d_{min} = \min_{e \in E(K_n)} d(e)$ et $d(T) = \sum_{e \in E(T)} d(e)$. Partant donc d'une instance I de MIN TSP, on construit une instance I' de MIN METRIC TSP en posant $\forall e \in E(K_n), d'(e) = d(e) + d_{max}$; on obtient alors pour tout cycle hamiltonien T la relation suivante :

$$d'(T) = d(T) + n \times d_{max} \qquad [1.3]$$

On vérifie aisément que la fonction d' satisfait l'inégalité triangulaire. Pour MAX TSP, en posant $\forall e \in E(K_n), d'(e) = d_{max} + d_{min} - d(e)$, la relation devient :

$$d'(T) = n \times (d_{max} + d_{min}) - d(T) \qquad [1.4]$$

En réalité, puisque l'on a trivialement les égalités $d'_{min} = d_{min}$ et $d'_{max} = d_{max}$, cette transformation de la fonction distance permet également d'obtenir la relation [1.4] pour la construction d'une instance $I' = (K_n, d')$ de MIN TSP à partir d'une instance $I = (K_n, d)$ de MAX TSP. De toutes ces relations, on déduit qu'un algorithme résolvant l'une de ces versions nous permet de déduire pour chacune d'elles un algorithme qui la résout et dont la complexité est du même ordre, à $O(n^2)$ près. On parle alors de *réduction polynomiale*, voir le chapitre 2 du volume 2 écrit, par G. Ausiello et V. Th. Paschos, c'est-à-dire que l'on ajoute à la complexité de l'algorithme initial au plus un polynôme pour construire de nouveaux algorithmes. Cette promiscuité qui lie ces trois versions du TSP du point de vue de leur résolution, nous l'explicitons formellement au travers de deux lemmes.

LEMME 1.1.– MIN TSP *et* MIN METRIC TSP *sont, en termes de complexité, équivalents à résoudre à l'ajout d'un facteur* $O(n^2)$ *près.*

Preuve. Soit `AlgoTSP` un algorithme exact pour MIN TSP ; puisque MIN METRIC TSP est un sous-problème de MIN TSP, l'algorithme `AlgoTSP` résout en particulier MIN METRIC TSP. Réciproquement, supposons que `AlgoTSP` soit un algorithme exact pour MIN METRIC TSP et voyons comment il pourrait nous permettre de résoudre MIN TSP. Soit $I = (K_n, d)$ une instance de MIN TSP, on construit l'instance $I' = (K_n, d')$ de MIN METRIC TSP où d' est définie par : $\forall e \in E(K_n), d'(e) = d(e) + d_{max}$. On lance alors `AlgoTSP` sur I' et l'on considère la solution renvoyée comme solution de l'instance initiale I. Or, l'égalité [1.3] nous permet d'affirmer qu'un cycle hamiltonien T^* est solution optimale sur I pour MIN TSP si et seulement s'il est solution optimale sur I' pour MIN METRIC TSP : nous déduisons ainsi qu'une telle procédure permettra de résoudre MIN TSP sachant que l'on sait résoudre MIN METRIC TSP, et ce en dégradant au plus d'un facteur $O(n^2)$ la complexité de `AlgoTSP`, puisque c'est le temps nécessaire à la construction de l'instance I'. ∎

Selon le même principe, l'égalité [1.4] et la remarque qui la suit nous permettent d'écrire et de justifier le lemme 1.2.

LEMME 1.2.– MIN TSP *et* MAX TSP *sont, en termes de complexité, équivalents à résoudre à un ajout d'un facteur* $O(n^2)$ *près.*

1.3. Algorithmes de résolution exacte

La première idée qui vient à l'esprit lorsque l'on veut résoudre TSP est certainement celle de la *recherche exhaustive*. Le principe en est très simple, mais au prix d'une complexité en temps très élevée : il consiste à déterminer toutes les solutions, à en évaluer la valeur, puis à sélectionner la meilleure de ces solutions. Dans notre contexte, cela se traduit par la recherche de tous les cycles hamiltoniens. Or, dans un graphe complet K_n, il y a $\frac{(n-1)!}{2}$ tours possibles ; sachant que l'évaluation d'un tour nécessite un temps $O(n)$, nous obtenons une complexité totale en temps de $O(n!)$, ce qui est équivalent, lorsque n tend vers l'infini, à $O(n^n\sqrt{2\pi n}e^{-n})$. Nous allons cependant voir que l'on peut diminuer de beaucoup cette complexité.

1.3.1. *Un algorithme de programmation dynamique*

Cet algorithme, simple à expliciter, utilise le principe de la *programmation dynamique*, voir le chapitre 4 du volume 1, écrit par B. Escoffier et O. Spanjaard. Il peut être décrit de la manière suivante, [HEL 62] : tout d'abord, trouver pour chaque sommet v_i du graphe une plus courte chaîne de v_1 à v_i qui visite tous les sommets de V ; ensuite, ajouter à chacune de ces chaînes l'arête retour qui permet de former un cycle ; enfin, choisir le meilleur des cycles ainsi construits.

Pour déterminer la meilleure chaîne de v_1 à v_i, l'algorithme résout le problème plus général qui consiste à déterminer pour tout $V' \subseteq V \backslash \{v_1\}$ et tout sommet v de V' la meilleure chaîne d'extrémités v_1 et v qui visite une et une seule fois chaque sommet de V'. Par la suite, on note par $\texttt{EnsembleCh}(V', v)$ l'ensemble des chaînes qui vérifient ces exigences, par $\texttt{MeilleurCh}(V', v)$ une plus courte chaîne parmi celles-ci et par $\texttt{ValeurCh}(V', v)$ sa valeur. On montre aisément que lorsque $|V'| \geq 2$, la quantité $\texttt{ValeurCh}(V', v)$ satisfait pour tout sommet v de V' : $\texttt{ValeurCh}(V', v) = \min_{w \neq v \in V'} \{\texttt{ValeurCh}(V' \setminus \{v\}, w) + d(w, v)\}$. Cette relation traduit la réflexion suivante : « sur V' et pour l'un de ses sommets v, il suffit, pour déterminer la plus courte chaîne de v_1 à v qui passe par chaque sommet de V', de connaître les plus courtes chaînes partant de v_1 qui passent par chaque sommet de $V' \setminus \{v\}$ ». L'algorithme consiste à exploiter cette relation : il construit itérativement $\texttt{MeilleurCh}(V', v)$ et $\texttt{ValeurCh}(V', v)$ pour des sous-ensembles V' de plus en plus grands, jusqu'a obtenir $\texttt{MeilleurCh}(\{v_2, \ldots, v_n\}, v)$ pour tout sommet v de $V \setminus \{v_1\}$.

Algorithme de programmation dynamique :

1) Pour $i = 2$ à n, faire :

2) $\quad \texttt{MeilleurCh}(\{v_i\}, v_i) = (v_1, v_i)$ et $\texttt{ValeurCh}(\{v_i\}, v_i) = d(v_1, v_i)$;

3) Pour $j = 2$ à $n - 1$, faire :

4) $\quad$ Pour tout $V' \subseteq \{v_2, \ldots, v_n\}$ de cardinalité j, faire :

5) $\quad$ Pour tout $v \in V'$, faire :

6) $\quad\quad \texttt{ValeurCh}(V', v) = \min_{w \in V'} \{\texttt{ValeurCh}(V' \setminus \{v\}, w) + d(w, v)\}$;

7) $\quad\quad \texttt{MeilleurCh}(V', v) = (\texttt{MeilleurCh}(V' \setminus \{v\}, w_0), v)$ (où w_0 est le sommet qui atteint ce minimum) ;

8) Renvoyer $T = \arg\min_{v=v_2}^{v_n} \{\texttt{ValeurCh}(\{v_2, \ldots, v_n\} \setminus \{v\}, v) + d(v, v_1)\}$.

La remarque précédente permet de conclure que cet algorithme construit bien une solution optimale de MIN TSP. Nous estimons maintenant sa complexité en temps, qui est essentiellement déterminée par les trois boucles imbriquées (étapes 3, 4 et 5). Fixons dans un premier temps l'entier $j \in \{2, \ldots, n-1\}$ (itération j de la boucle 3), l'ensemble $V' \subseteq \{v_2, \ldots, v_n\}$ de taille j (itération V' de la boucle 4) et le sommet $v \in V'$ (itération v de la boucle 5) ; trouver $\texttt{ValeurCh}(V', v)$ et $\texttt{MeilleurCh}(V', v)$ nécessite un temps $O(j)$ connaissant les quantités $\texttt{ValeurCh}(V' \setminus \{v\}, w)$ pour $w \neq v \in V'$. Puisque cette opération est répétée pour chaque sommets v de V', l'itération V' de la boucle 5 prend globalement un temps $O(j^2)$. Cette boucle étant elle-même répétée pour chaque sous-ensemble V' de taille j parmi $\{v_2, \ldots, v_n\}$, elle consomme en tout un temps $C_{n-1}^j O(j^2)$, où l'on rappelle que C_n^j vaut $\frac{n!}{j!(n-j)!}$. Enfin, l'étape 3 est déroulée pour chaque $j \in \{2, \ldots, n-1\}$, ce qui nous permet de déduire une complexité totale en temps majorée par la quantité $\sum_{j=2}^{n-1} C_{n-1}^j O(j^2) = O(n^2 2^n)$.

THÉORÈME 1.1.– *[HEL 62] L'algorithme de programmation dynamique résout optimalement* MIN TSP.

1.3.2. *Un algorithme de séparation et évaluation*

L'algorithme, conçu par J.D.C. Little, K.G. Murty, D.W. Sweeney et C. Karel, voir [LIT 63], met en œuvre une méthode par séparation et évaluation basée sur l'heuristique du regret. Il s'agit maintenant de résoudre le TSP dans un cadre plus général : sa version asymétrique. Nous travaillons sur un graphe $G = (V, A)$ complet orienté sur n sommets $\{v_1, \ldots, v_n\}$ où la distance $d(v_i, v_j)$ pour se rendre de v_i à v_j n'est pas nécessairement la même que la distance $d(v_j, v_i)$ de l'arc (v_j, v_i). Notons que pour transformer une instance symétrique en une instance asymétrique, il suffit de dupliquer chaque arête $[v_i, v_j]$ en deux arcs (v_i, v_j) et (v_j, v_i) de même distance que l'arête initiale ; cette procédure, qui nécessite un temps $O(n^2)$, permet d'appliquer un algorithme conçu pour le cas asymétrique à toute instance du cas symétrique.

Le regret est la façon d'évaluer le coût, que l'on pourrait qualifier *d'inopportunité*, de la non-incorporation d'un arc à la solution ; pour le calculer, les distances doivent au préalable subir une opération appelée *réduction*. On commence par retirer à chaque arc $\vec{a} = (v_i, v_j)$ le plus petit coût pour partir de i, soit formellement : $\forall i = 1, \ldots, n$, on note $D^i = \min_{j \neq i}\{d(v_i, v_j)\}$ et l'on pose, pour tout $j \neq i$, $d'(v_i, v_j) = d(v_i, v_j) - D^i$. C'est ensuite le plus petit coût pour arriver en i et relativement à d' que l'on retire : $\forall j = 1, \ldots, n$, on note $D'^j = \min_{i \neq j}\{d'(v_i, v_j)\}$ et l'on pose $d''(v_i, v_j) = d'(v_i, v_j) - D'^j$ pour tout $i \neq j$. Il est aisé de constater que la fonction d'' ne transforme pas l'ordre relatif des tours réalisables ; en effet, si l'on note $D' = \sum_{i=1}^{n} D^i$, $D'' = \sum_{j=1}^{n} D'^j$ et $D = D' + D''$, alors les fonctions d, d' et d'' vérifient la double inégalité : $\forall T,\ d''(T) = d'(T) - D'' = d(T) - D$. La figure 1.1 déroule l'étape de réduction sur un graphe à cinq sommets.

$D' = 10$

	v_1	v_2	v_3	v_4	v_5	
v_1	-	11	1	7	9	1
v_2	5	-	3	12	3	3
v_3	7	1	-	9	13	1
v_4	14	9	5	-	4	4
v_5	3	12	7	1	-	1
						10

$D'' = 2$

	v_1	v_2	v_3	v_4	v_5	
v_1	-	10	0	6	8	
v_2	2	-	0	9	0	
v_3	6	0	-	8	12	
v_4	10	5	1	-	0	
v_5	2	11	6	0	-	
	2					**2**

$D = 12$

	v_1	v_2	v_3	v_4	v_5
v_1	-	10	0^6	6	8
v_2	0^0	-	0^0	9	0^0
v_3	4	$\mathbf{0}^9$	-	8	12
v_4	8	5	1	-	0^1
v_5	0^0	11	6	0^6	-

Figure 1.1. *Calcul des coûts réduits sur le graphe initial*

La constante D offre une première estimation du coût d'une solution optimale sur G ($\forall T$, $d(T) = d''(T) + D$ et $d''(T) \geqslant 0 \Rightarrow d(T) \geqslant D$). De surcroît, la réduction

permet de faire apparaître des arcs de coût nul partant de chaque sommet et arrivant en chaque sommet ; c'est précisément sur ces arcs que les regrets sont calculés, traduisant l'arbitrage suivant : « Si incorporer un arc de coût nul à la solution n'a aucune incidence sur l'évaluation de celle-ci, combien nous en coûtera-t-il de ne pas l'emprunter ? ». Sur le dernier tableau de la figure 1.1, considérons l'arc (v_3, v_2) : l'emprunter ne coûterait rien, tandis que l'interdire imputerait à la solution un coût supplémentaire d'au moins 4 (coût minimal pour partir de v_3 autrement qu'en allant vers v_2) + 5 (coût minimal pour arriver en v_3 autrement que par v_2) = 9 : c'est la notion de regret. Formellement, le calcul des regrets revient à déterminer pour tout arc $\vec{a} = (v_i, v_j)$ tel que $d''(\vec{a}) = 0$ la quantité : $r(\vec{a}) = \min_{k \neq j,i}\{d(v_i, v_k)\} + \min_{k \neq i,j}\{d(v_k, v_j)\}$. Sur le dernier tableau de la figure 1.1, les regrets sont indiqués en exposant des arcs de coût 0.

Pour exploiter l'outil de réduction et de calcul des regrets dans le cadre d'une procédure d'exploration par séparation et évaluation voir le chapitre 3 du volume 1 écrit par I. Charon et O.Hudry, l'algorithme de Little sépare l'ensemble $\mathcal{S}$ des solutions réalisables en deux sous-ensembles complémentaires, respectivement qualifiés de nœuds *d'inclusion* et *d'exclusion* : les solutions du premier empruntent l'arc de plus grand regret, au contraire des solutions du second. En fonction de l'opportunité qu'il représente, un sous-ensemble S sera à son tour séparé, toujours selon le même arbitrage, et ainsi de suite. Les feuilles de l'arborescence d'exploration seront soit des ensembles réduits à une solution unique, soit des nœuds stérilisés, c'est-à-dire dont on sait, à un moment donné du déroulement de l'algorithme, qu'ils ne contiennent pas de solution qui soit meilleure que la meilleure solution connue.

Un ensemble S de solutions est décrit par les ensembles P_S et M_S des arcs de plus grand regret qui ont été successivement imposés (P_S) ou interdits (M_S) aux solutions de S. La fonction des coûts sur S est notée d_S et résulte des réductions successives de la fonction d initiale ; on désigne par d''_S sa version réduite, par D_S la différence entre les évaluations faites par d_S et d''_S d'un tour de S. Enfin, on se munit d'une évaluation minimale des solutions de S, notée $b(S)$: pour évaluer un nœud S, on fait la somme de l'évaluation de son père et de la constante D_S ; lorsque S est un nœud d'exclusion, D_S est la valeur du plus grand regret sur son père. Notons que l'ensemble A_S des arcs valides sur S, indépendamment de l'ensemble M_S, évolue à chaque fois qu'un arc est imposé aux ensembles de solutions : en plus des arcs de M_S, l'ensemble A_S (qui est égal à A pour $S = \mathcal{S}$) exclut tous les arcs (v_i, v_j) tels que $v_i \in \Gamma^-(P_S)$ ou $v_j \in \Gamma^+(P_S)$, ainsi que tout arc $\vec{a}$ qui, ajouté à P_S, créerait un sous-circuit[1]. La présentation de l'algorithme est organisée en deux parties : la première détaille les opérations qui sont faites sur un nœud, c'est-à-dire la réduction des coûts,

1. Soit A un ensemble d'arcs, $\Gamma^-(A)$ (resp., $\Gamma^+(A)$) désigne l'ensemble des sommets qui sont extrémité initiale (resp., finale) d'un arc de A.

la détermination de l'arc de plus grand regret et la séparation, tandis que la seconde traite de la stratégie globale d'exploration.

Réduction des coûts :

1) Poser $D'_S = D''_S = 0$;

2) Pour $i = 1, \ldots, n$ tel que $i \notin \Gamma^-(P_S)$, faire :

3) poser $D^i_S = \min\{d_S(\vec{a}) \mid \vec{a} = (v_i, v_j) \in A_S\}$ et $D'_S = D'_S + D^i_S$;

4) pour tout $\vec{a} = (v_i, v_j) \in A_S$, poser $d'_S(\vec{a}) = d''_S(\vec{a}) - D^i_S$;

5) Pour $= 1, \ldots, n$ tel que $j \notin \Gamma^+(P_S)$, faire :

6) poser $D'_S{}^j = \min\{d'_S(\vec{a}) \mid \vec{a} = (v_i, v_j) \in A_S\}$ et $D''_S = D''_S + D'_S{}^j$;

7) pour tout $\vec{a} = (v_i, v_j) \in A_S$, poser $d''_S(\vec{a}) = d'_S(\vec{a}) - D'_S{}^j$;

8) Poser $D_S = D'_S + D''_S$.

Arc de plus grand regret :

1) Pour tout $\vec{a} = (v_i, v_j) \in A_S$ / $d''_S(\vec{a}) = 0$, poser
$r(\vec{a}) = \min\limits_{k \neq j}\{d''_S(\vec{a}) \mid \vec{a} = (v_i, v_k) \in A_S\} + \min\limits_{k \neq i}\{d''_S(\vec{a}) \mid \vec{a} = (v_k, v_j) \in A_S\}$;

2) Renvoyer $\vec{a}_S$ l'arc de regret maximal parmi les arcs de coût d''_S nul.

Séparation :

1) Si $S = \mathcal{S}$, réduire $d_{\mathcal{S}}$ puis poser $b(\mathcal{S}) = D_{\mathcal{S}}$;

2) Sinon, si S est un nœud d'exclusion, réduire d_S ;

3) Déterminer l'arc $\vec{a}_S$ de plus grand regret puis séparer S en S_P et S_M ;

4) Sur S_P :

5) poser $P_{S_P} = P_S \cup \{\vec{a}_S\}$ et $M_{S_P} = M_S$;

6) réduire d_S puis poser $b(S_P) = b(S) + D_S$;

7) Sur S_M, poser $P_{S_M} = P_S$, $M_{S_M} = M_S \cup \{\vec{a}_S\}$, $b(S_M) = b(S) + r(\vec{a}_S)$.

La stratégie d'exploration choisie est celle de l'exploration *en profondeur d'abord*, ne serait-ce que pour disposer rapidement d'une solution et par conséquent, d'une borne supérieure de la valeur d'une solution optimale. En effet, le but est de limiter au mieux le développement de l'arborescence et de telles évaluations, parce qu'elles permettent de stériliser des nœuds, constituent un outil précieux. Mais si l'on persiste dans cette voie, c'est surtout par soucis de cohérence avec la méthode de séparation. L'évaluation des nœuds peut être hétérogène, si certains sont une succession d'incorporations quand d'autres sont une succession de refus : un coût plus important sur une solution à moitié dessinée peut néanmoins sembler préférable à un coût moindre sur une solution dénuée d'arcs. Dans la description qui suit de l'algorithme, $\overline{b}$ décrit la meilleure borne supérieure connue pour la valeur d'un tour optimal.

Algorithme de séparation et évaluation :

1) Poser $b(\mathcal{S}) = 0$, $d_{\mathcal{S}} = d$, $P_{\mathcal{S}} = M_{\mathcal{S}} = \emptyset$ et $\overline{b} = +\infty$;

2) Tant qu'il existe une feuille S telle que $b(S) < \overline{b}$:

3) choisir S *t.q.* $b(S)$ est minimal parmi les feuilles *t.q.* $|P_S|$ est maximal ;

4) si S est réduite à un tour poser $\overline{b} = \min\{\overline{b}, b(S)\}$, sinon, séparer S ;

5) stériliser tout nœud S tel que $b(S) > \overline{b}$;

6) Renvoyer la feuille S réduite à une solution et telle que $b(S) = \overline{b}$.

Nous reprenons maintenant le cours de notre exemple ; le lecteur est invité à suivre le déroulement de l'arborescence sur la figure 1.4. Le nœud initial, noté S_0, admet pour arc de plus grand regret l'arc (v_3, v_2), de regret 9 ; on le sépare donc en S_1 (solutions issues de S_0 qui empruntent (v_3, v_2)) et S_2 (solutions issues de S_0 qui n'empruntent pas (v_3, v_2)), puis l'on pose $b(S_2) = b(S_0) + 9 = 21$. Le fait d'intégrer (v_3, v_2) aux tours de S_1 rend prohibés l'arc (v_2, v_3), les arcs (autres que (v_3, v_2)) partant de v_3 et les arcs (autres que (v_3, v_2)) arrivant en v_2.

$\vec{a}_{S_1} = (v_1, v_3)$	$\vec{a}_{S_3} = (v_2, v_5)$	$D'_{S_5} = 8$	$S_5 = \{(v_1 v_3 v_2 v_5 v_4 v_1)\}$
v_1 - **0^7** 6 8	v_2 - 9 **0^9**	v_4 8 - | 8	v_4 **0** -
v_2 0^0 - 9 0^0	v_4 8 - 0^8	v_5 - 0 |	v_5 **0**
v_4 8 1 - 0^1	v_5 0^8 0^9 -	**8**	
v_5 0^0 6 0^6 -			

Figure 1.2. *Première solution*

Les coûts sur S_1 (premier tableau de la figure 1.2) sont déjà sous forme réduite ; on pose alors $b(S_1) = b(S_0) = 12$. Conformément à la stratégie d'exploration ($|P_{S_1}| = 1$ contre $|P_{S_2}| = 0$), le nœud courant devient S_1 et l'arc de décision (v_1, v_3), de regret 7. S_1 est séparé en S_3 (qui impose (v_1, v_3) mais interdit (v_2, v_1) afin de ne pas générer le sous-circuit $(v_1 v_3 v_2 v_1)$) et S_4 (qui refuse (v_1, v_3)) ; ces ensembles sont respectivement évalués à $b(S_4) = 19$ et $b(S_3) = 12$. Le nouveau nœud d'étude est alors S_3, sur lequel l'arc (v_2, v_5) est de regret maximal (deuxième tableau de la figure 1.2). S_3 est à son tour séparé en S_5 et S_6, où S_5 regroupe les solutions passant par (v_1, v_3), (v_3, v_2), (v_2, v_5) et S_6 celles qui passent par (v_1, v_3), (v_3, v_2) et non (v_2, v_5) ; on pose par ailleurs $b(S_6) = 21$. La réduction des coûts sur S_5 nous donne $D_S = D'_S = 8$, dont on déduit $b(S_5) = b(S_3) + 8 = 20$; le nœud S_5 devient le nœud courant. Finalement, comme le montre le troisième tableau, S_5 décrit une solution unique, $(v_1 v_3 v_2 v_5 v_4 v_1)$, de valeur 20. On pose alors $\overline{b} = 20$ et l'on stérilise les feuilles qui ne peuvent promettre mieux que 20 : c'est le cas de S_2 et S_6.

$D'_{S_4} = 6$

	v_1	v_3	v_4	v_5	
v_1	-	-	6	8	6
v_2	0	-	9	0	
v_4	8	1	-	0	
v_5	0	6	0	-	
					6

$D''_{S_4} = 1$

	v_1	v_3	v_4	v_5
v_1	-	-	0	2
v_2	0	-	9	0
v_4	8	1	-	0
v_5	0	6	0	-
		1		**1**

$\vec{a}_{S_4} = (v_4, v_3)$

	v_1	v_3	v_4	v_5
v_1	-	-	0^2	2
v_2	0^0	-	9	0
v_4	8	$\mathbf{0}^5$	-	0^0
v_5	0^0	5	0^0	-

Figure 1.3. *Réduction des coûts et calcul des regrets sur S_4*

La seule feuille disponible est S_4 ; puisqu'il s'agit d'un nœud d'exclusion, il n'est pas nécessaire de réévaluer $b(S_4)$: la constante de réduction a déjà été incorporée par l'intermédiaire du regret de ne pas emprunter (v_1, v_3). Les regrets sur S_4 (dernier tableau de la figure 1.3) nous indiquent que c'est l'arc (v_4, v_3) qui sépare S_4 en S_7 et S_8. Les solutions de S_8 coûtent au moins 24, celles de S_7 au moins 19. Le nœud S_8, à peine créé, est déjà stérilisé, tandis que S_7 est mis à l'étude (deuxième tableau de la figure 1.3). Ce dernier nœud donne naissance à deux fils, S_9 (tours de S_7 empruntant (v_1v_4)) et $S10$ (tours de S_7 n'empruntant pas (v_1v_4)). Le nœud S_{10} est évalué à 21, il sera donc stérilisé. En revanche, S_9 décrit la solution unique $(v_1v_4v_3v_2v_5v_1)$, de valeur 19 : c'est une solution optimale (toutes les autres feuilles sont stérilisées) et unique (les évaluations faites des autres feuilles sont strictement pires que 19).

THÉORÈME 1.2.– *[LIT 63] L'algorithme de séparation et évaluation résout* MIN TSP *à l'optimum.*

1.4. Algorithme approché pour max TSP

Nous présentons ici deux algorithmes approximant MAX TSP suivant deux ratios de performance : le *rapport standard* et le *rapport différentiel*. Ces ratios traduisent l'existence d'un réel $r \in]0\,;1]$ vérifiant, pour toute instance I, $val(T) \geq r \times opt_{maxTSP}(I)$ pour le premier, $val(T) \geq r \times opt_{maxTSP}(I) + (1-r)wor_{maxTSP}(I)$ pour le second, où $opt_{maxTSP}(I)$, $wor_{maxTSP}(I)$ et $val(T)$ désignent respectivement la valeur d'un *tour optimal*, d'un *plus mauvais tour* sur l'instance I et d'un *tour approché*. Concrètement, il s'agit de situer la valeur de la solution approchée relativement à la valeur optimale pour le rapport classique, relativement aux valeurs d'une pire et d'une meilleure solutions en différentiel : quitte à utiliser des algorithmes approchés, on souhaite pour le moins estimer la qualité des solutions fournies, en situant leur valeur sur le spectre des valeurs possibles. Le schéma 1.5 illustre les mesures classique et différentielle pour un problème de maximisation en indiquant les intervalles discrets pour des solutions 1/2-approchées relativement aux deux rapports ; la valeur d'une pire solution est notée $wor(I)$, la valeur optimale est notée $opt(I)$.

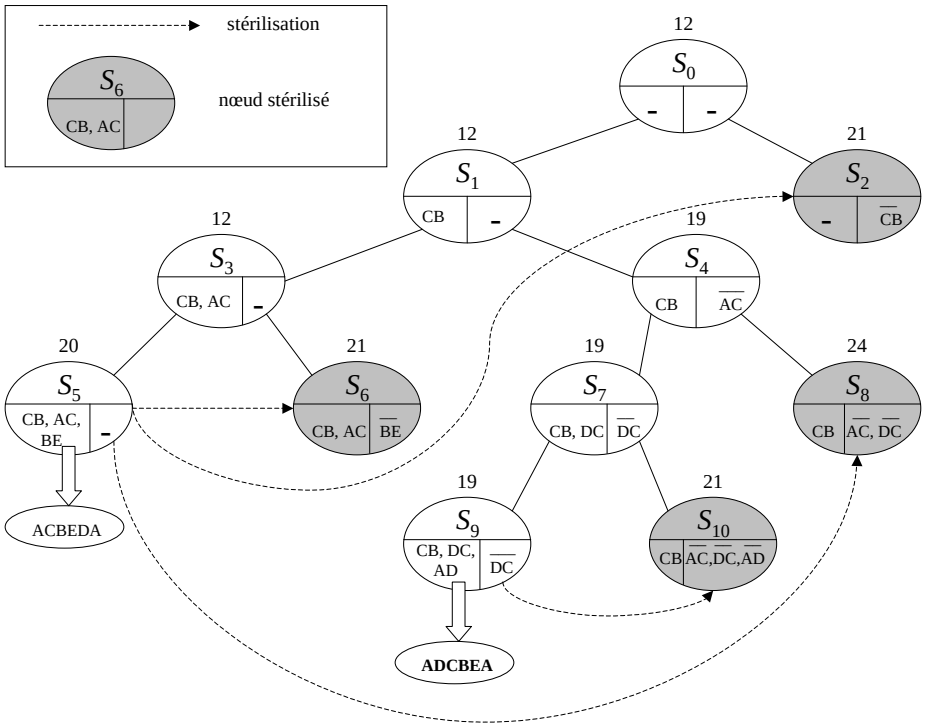

Figure 1.4. *Algorithme de Little*

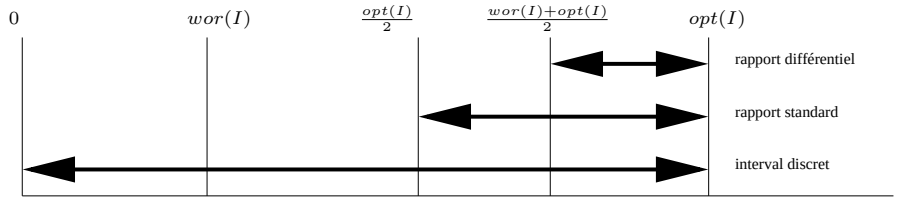

Figure 1.5. *Positionnement d'une solution $\frac{1}{2}$-approchée*

Pour connaître en détail les concepts liés à la notion d'algorithme à garantie de performances, le lecteur est invité à consulter le chapitre 1 du volume 2 écrit par M. Demange et V. Th. Paschos.

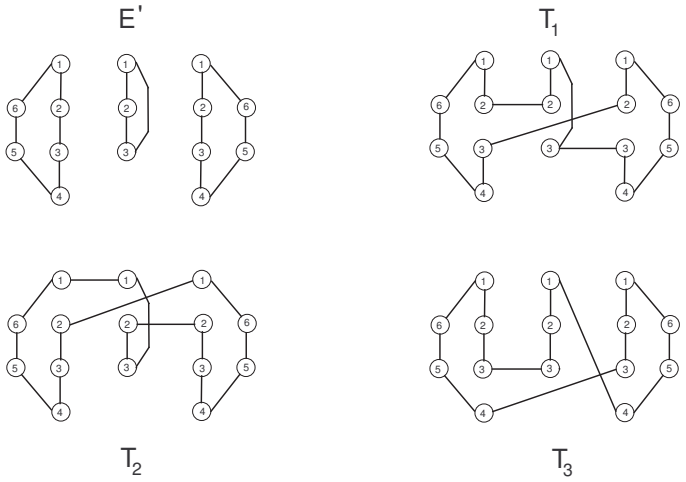

Figure 1.6. *Illustration du fonctionnement de l'algorithme lorsque* $k = 3$

La première voie de résolution consiste à adapter l'idée proposée pour la mesure standard dans [FIS 79] où les auteurs, partant d'un 2-couplage parfait de poids maximal, modifient celui-ci de manière à obtenir un tour. Un ensemble d'arêtes $E' \subseteq E$ d'un graphe $G = (V, E)$ est un *2-couplage* si chaque sommet de G est incident à au plus 2 arêtes de E' ; notons que c'est le cas d'un cycle hamiltonien, puisque dans un cycle, tout sommet est de degré exactement 2. Plus généralement, supposons que soit affecté à chaque sommet v un entier naturel $f(v)$: un *f-couplage* est alors un ensemble d'arêtes $E' \subseteq E$ qui vérifie que chaque sommet v est incident à au plus $f(v)$ arêtes de E'. Lorsque chaque sommet v est incident à exactement $f(v)$ arêtes de E', le f-couplage est dit *parfait*. Un autre cas particulier de f-couplage que l'on rencontre souvent est celui où la fonction $f(v)$ est à valeur dans $\{0, 1\}$: on parle alors de *couplage*. Finalement, on appelle *f-couplage de poids maximal* un f-couplage E' dont la somme des poids des arêtes est maximale. Dans [HAR 84], il est proposé un algorithme de complexité $O(n^3)$ qui construit un *2-couplage de poids maximal* dans un graphe donné. Par le biais d'un léger artifice, cet algorithme permet également de construire un *2-couplage parfait de poids maximal* : il suffit pour ce faire d'augmenter le poids de chaque arête de telle sorte que chaque 2-couplage de poids maximal soit nécessairement parfait. Pour la description des algorithmes construisant un f-couplage, voir [GON 95].

Les problèmes de 2-couplage se complexifient néanmoins lorsque l'on ajoute une certaine contrainte quant à la forme des cycles, plus précisément sur le nombre minimal de sommets que les cycles du couplage doivent contenir : on parle alors de

2-couplage k-restreint, voir [KIR 99]. Un *2-couplage parfait* est dit k-restreint si tous ses cycles contiennent au moins $k+1$ sommets. Il est remarquable qu'un 2-couplage parfait est 2-restreint (chaque cycle contient au moins 3 sommets) ou encore qu'un 2-couplage parfait $n-1$-restreint est un cycle hamiltonien ; en réalité, on peut prouver qu'un 2-couplage parfait k-restreint est un cycle hamiltonien si et seulement si $\frac{n}{2} \leq k \leq n-1$. Le problème de déterminer un 2-couplage parfait 4-restreint de poids maximal est **NP**-difficile, voir [PAP 98, VOR 80]), tandis que le statut du problème consistant à déterminer un 2-couplage parfait 3-restreint (c'est-à-dire un 2-couplage parfait sans triangle) qui soit de poids maximal est une question toujours ouverte à l'heure actuelle. Cependant, lorsque le graphe est non pondéré, il existe un algorithme polynomial pour construire un 2-couplage parfait 3-restreint, un autre pour construire un 2-couplage parfait 4-restreint si le graphe est biparti, voir [HAR 99] (sous la condition, bien sûr, que de tels 2-couplages existent).

1.4.1. *Un algorithme basé sur le 2-couplage*

Puisqu'un tour optimal est un 2-couplage parfait particulier (ne contenant qu'un cycle), ces deux notions sont très proches. Aussi, les auteurs de [FIS 79] ont conçu leur algorithme de la manière suivante : pour chaque cycle du 2-couplage parfait de poids maximal, on supprime une arête de plus petit coût, puis on relie aléatoirement les chaînes résultantes de manière à obtenir un cycle hamiltonien. Le procédé est assez simple, aussi est-il laissé au lecteur le soin de démontrer que cet algorithme, dont la complexité en temps est de $O(n^3)$, retourne un tour T qui vérifie : $\forall I,\ val(T) \geq \frac{2}{3} opt_{maxTSP}(I)$. Malheureusement, dans le pire des cas, la solution T de cet algorithme ne garantit pas l'inégalité $val(T) \geq \frac{2}{3} opt_{maxTSP}(I) + \frac{1}{3} wor_{maxTSP}(I)$: cela signifie que ce tour peut être beaucoup plus proche de la pire solution que de l'optimum.

L'algorithme que nous présentons maintenant est issu de [MON 02b]. Il consiste à générer, non plus un, mais plusieurs cycles hamiltoniens, chacun d'eux étant construit selon la méthode décrite précédemment. L'idée générale est la suivante : partant d'une collection de cycles qui couvre les sommets de K_n, chaque cycle étant de longueur au moins 3, on crée trois cycles hamiltoniens, le $i$ème cycle hamiltonien consistant à retirer la $i$ème arête de chaque cycle du 2-couplage puis à relier les cycles ainsi amputés entre eux de sorte à former un cycle unique. Plus rigoureusment, soit E' un 2-couplage parfait de poids maximal décrit par les cycles $C_i,\ i = 1, \ldots, k$; pour chacun de ces cycles C_i, nous considérons 4 sommets consécutifs v_1^i, v_2^i, v_3^i et v_4^i (en particulier, lorsque $|C_i| = 3$, nous avons $v_4^i = v_1^i$) ; finalement, pour le dernier cycle C_k, nous considérons également l'autre voisin dans C_k de v_1^k, que l'on note y. Notons que si $|C_k| = 3$, les sommets y et v_3^k coïncident, tandis que si $|C_k| = 4$, ce sont les sommets y et v_4^k qui sont un seul et même sommet ; autrement, y est toujours un nouveau sommet.

Algorithme basé sur le 2-couplage :

1) Construire un 2-couplage parfait E' de poids maximal dans $I = (K_n, d)$ contenant l'ensemble de cycles $\{C_1, \ldots, C_k\}$;

2) Si $k = 1$, alors poser $E(T) = E'$ et renvoyer T ;

3) Si k est impair, alors faire :

4) Poser $S_1 = \cup_{j=1}^{k}\{[v_2^j, v_3^j]\}$ et construire $E(T_1) = (E' \setminus S_1) \cup \{[v_2^k, v_3^1]\} \cup_{j=1}^{(k-1)/2} \{[v_2^{2j-1}, v_2^{2j}], [v_3^{2j}, v_3^{2j+1}]\}$;

5) Poser $S_2 = \cup_{j=1}^{k}\{[v_1^j, v_2^j]\}$ et construire $E(T_2) = (E' \setminus S_2) \cup \{[v_1^k, v_2^1]\} \cup_{j=1}^{(k-1)/2} \{[v_1^{2j-1}, v_1^{2j}], [v_2^{2j}, v_2^{2j+1}]\}$;

6) Poser $S_3 = \cup_{j=1}^{k}\{[v_3^j, v_4^j]\}$ et construire $E(T_3) = (E' \setminus S_3) \cup \{[v_3^k, v_4^1]\} \cup_{j=1}^{(k-1)/2} \{[v_3^{2j-1}, v_3^{2j}], [v_4^{2j}, v_4^{2j+1}]\}$;

7) Si k est pair, alors faire :

8) Poser $S_1 = \cup_{j=1}^{k-1}\{[v_2^j, v_3^j]\} \cup \{[v_1^k, v_2^k]\}$ et construire $E(T_1) = (E' \setminus S_1) \cup \{[v_1^k, v_3^1], [v_2^1, v_2^2]\} \cup_{j=1}^{(k-2)/2} \{[v_3^{2j}, v_3^{2j+1}], [v_2^{2j+1}, v_2^{2j+2}]\}$;

9) Poser $S_2 = \cup_{j=1}^{k-1}\{[v_1^j, v_2^j]\} \cup \{[y, v_1^k]\}$ et construire $E(T_2) = (E' \setminus S_2) \cup \{[y, v_2^1], [v_1^1, v_1^2]\} \cup_{j=1}^{(k-2)/2} \{[v_2^{2j}, v_2^{2j+1}], [v_1^{2j+1}, v_1^{2j+2}]\}$;

10) Poser $S_3 = \cup_{j=1}^{k-1}\{[v_3^j, v_4^j]\} \cup \{[v_2^k, v_3^k]\}$ et construire $E(T_3) = (E' \setminus S_3) \cup \{[v_2^k, v_4^1], [v_3^1, v_3^2]\} \cup_{j=1}^{(k-2)/2} \{[v_4^{2j}, v_4^{2j+1}], [v_3^{2j+1}, v_3^{2j+2}]\}$;

11) Renvoyer $T = \arg\max\{val(T_1), val(T_2), val(T_3)\}$.

Il est aisé de voir que dans chacun des cas traités, les arêtes de l'ensemble $E(T_i)$ forment un cycle hamiltonien, de sorte que l'algorithme produit bien un tour. La figure 1.6 décrit le fonctionnement de cet algorithme lorsque $k = 3$. La complexité de cet algorithme est dominée par l'étape 1 qui consomme, d'après les remarques précédentes, un temps $O(n^3)$.

THÉORÈME 1.3.– *[MON 02b] Pour toute instance $I = (K_n, d)$, l'inégalité suivante est valable : $val(T) \geq \frac{2}{3} opt_{maxTSP}(I) + \frac{1}{3} wor_{maxTSP}(I)$.*

Preuve. Soient $I = (K_n, d)$ une instance du voyageur de commerce et T^* un tour optimal de valeur $opt_{maxTSP}(I)$ sur I. On note $perte_i = val(T_i) - d(E')$ pour $i = 1, 2, 3$; cette quantité est négative et représente le prix de la transformation du 2-couplage parfait E' en le tour T_i. La solution T retournée par l'algorithme est de valeur supérieure ou égale à la valeur de chaque tour T_i et ainsi, au moins aussi bonne que la moyenne des valeurs de ces tours. Autrement dit :

$$val(T) \geq \frac{1}{3}(\sum_{i=1}^{3} val(T_i)) = d(E') + \frac{1}{3}(perte_1 + perte_2 + perte_3) \qquad [1.5]$$

D'un autre coté, nous pouvons facilement vérifier, par un de ses parcours, que la solution T_* constituée par $E(T_*) = \cup_{j=1,2,3}(E(T_j) \setminus E') \cup (E' \setminus (S_1 \cup S_2 \cup S_3))$ est un cycle hamiltonien sur I de valeur $d(T_*) = d(E') + perte_1 + perte_2 + perte_3$. La figure 1.8 décrit ce cycle hamiltonien pour l'exemple illustré par la figure 1.7. Par définition d'une plus mauvaise solution, nous avons $wor_{maxTSP}(I) \leq d(T_*)$, ce qui revient à :

$$wor_{maxTSP}(I) \leq d(E') + perte_1 + perte_2 + perte_3 \quad [1.6]$$

Evaluons à présent la solution optimale T^* ; tout tour étant un 2-couplage parfait de I, cela est également vrai de T^* et l'on déduit :

$$d(E') \geq opt_{maxTSP}(I) \quad [1.7]$$

Finalement, en rapprochant les inégalités [1.5], [1.6] et [1.7], nous obtenons :

$$val(T) \geq \frac{2}{3} opt_{maxTSP}(I) + \frac{1}{3} wor_{maxTSP}(I) \quad [1.8]$$

Nous laissons au lecteur le soin de prouver que l'inégalité [1.8] est la meilleure possible pour cet algorithme, et ce même lorsque $d(e) \in \{1, 2\}$; pour prouver un tel fait, il suffit d'exhiber une famille d'instances qui atteint le rapport d'approximation énoncé pour l'algorithme soumis à l'analyse. Précisons pour conclure qu'un rapport semblable a été obtenu dans [HAS 01] et que cette inégalité [1.8] constitue à l'heure actuelle le meilleur rapport différentiel connu pour MAX TSP. ∎

1.4.2. *Algorithme mêlant 2-couplage et couplage*

Nous présentons ici un algorithme exposé dans [SER 84] et qui approxime MAX TSP relativement à la mesure standard. Afin de faciliter la compréhension de la preuve, nous avons volontairement modifié l'algorithme en réduisant son traitement principal et son analyse au cas où l'ordre n du graphe est impair. La contre-partie de cette transformation est l'augmentation de la complexité en temps d'un facteur $O(n^2)$. Le principe consiste à construire deux tours partiels $\mathcal{P}_1$ et $\mathcal{P}_2$ (un tour partiel est un ensemble de chaînes de K_n qui sont sommet-disjointes), de sélectionner le meilleur des deux, puis de prolonger celui-ci en un tour complet. Dans le but de garantir une bonne approximation de l'optimum, $\mathcal{P}_1$ et $\mathcal{P}_2$ sont obtenus à partir de deux bornes inférieures de l'optimum : le *2-couplage parfait de poids maximal* et le *couplage parfait de poids maximal*.

Algorithme meilleur tour partiel :

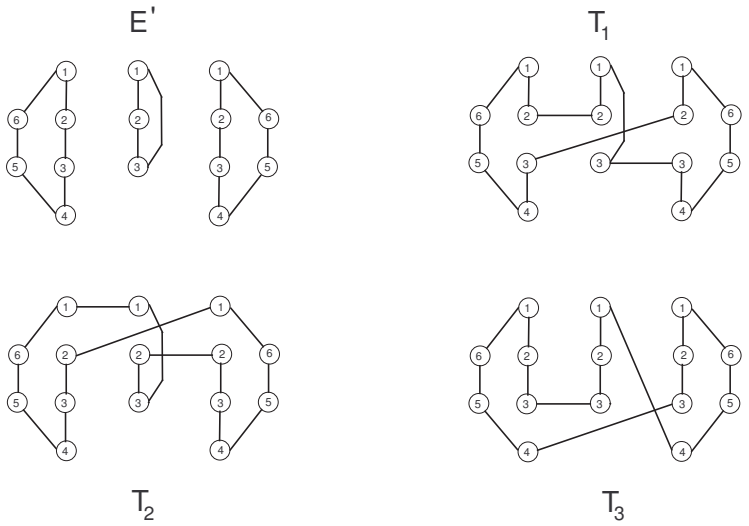

Figure 1.7. *Illustration du fonctionnement de l'algorithme lorsque* $k = 3$

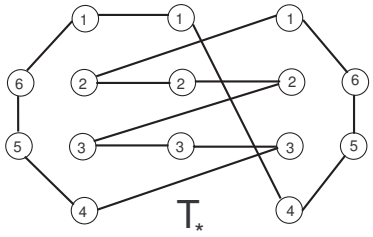

Figure 1.8. *La solution* T_*

1) Construire un couplage parfait de poids maximal E_1 dans I ;

2) Construire un 2-couplage parfait E_2 de poids maximal dans $I = (K_n, d)$, décrit par les cycles $C_1, \ldots, C_p$;

3) Poser $\mathcal{P}_1 = E_1$ et $\mathcal{P}_2 = \emptyset$;

4) Pour $i = 1$ à p, faire :

5) Trouver $e \in E(C_i)$ tel que $(\mathcal{P}_1 \cup \{e\})$ reste un tour partiel ;

6) Poser $\mathcal{P}_1 \leftarrow \mathcal{P}_1 \cup \{e\}$ et $\mathcal{P}_2 \leftarrow \mathcal{P}_2 \cup (E(C_i) \setminus \{e\})$;

7) Renvoyer $\mathcal{P} = argmax\{val(\mathcal{P}_1)\,; val(\mathcal{P}_2)\}$.

Cet algorithme est bien entendu d'exécution polynomiale en temps et sa complexité, d'ordre $O(n^3)$, imputable à la construction du 2-couplage parfait et du couplage parfait de poids maximal. La validité de cet algorithme est entièrement conditionnée par celle de l'étape 5 au sein de la boucle 4 (une illustration de la construction de $\mathcal{P}_1$ et $\mathcal{P}_2$ qui en découle est proposée par la figure 1.9). Pour l'établir, nous allons tout d'abord montrer par récurrence sur le numéro d'itération de la boucle 4 que nous avons les propriétés suivantes :

– chaque chaîne P de $\mathcal{P}_1$ (à l'étape courante) alterne arêtes de E_1 et de E_2 ;

– $\mathcal{P}_1$ (à l'étape courante) est un tour partiel.

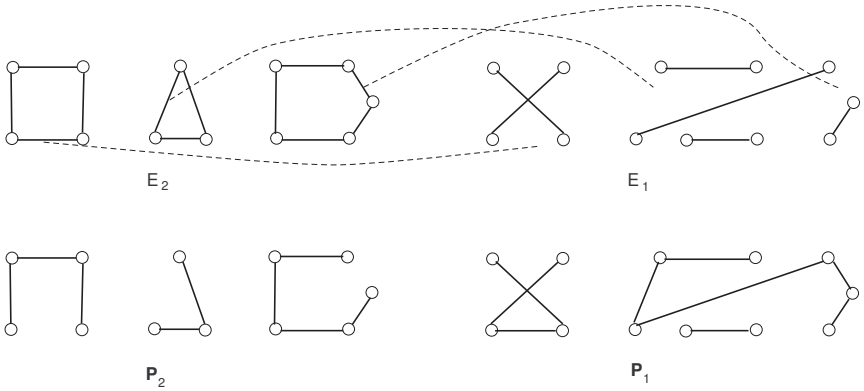

Figure 1.9. *Les tours partiels $\mathcal{P}_1$ et $\mathcal{P}_2$*

Pour $i = 0$, le résultat est vrai d'après l'étape 3 ; supposons que le résultat soit vrai à l'étape i pour i tel que $0 \leq i < p$ et prouvons qu'il l'est encore à l'étape $i + 1$. Considérant un sommet quelconque $v \in V(C_{i+1})$, on note par e_1 et e_2 les deux arêtes qui lui sont incidentes dans C_{i+1} et par P la chaîne de $\mathcal{P}_1$ qui passe par v. Si v est sommet interne de P, il est alors, par hypothèse de récurrence quant à la forme des chaînes de $\mathcal{P}_1$, adjacent à une arête de E_2 ; une telle situation induirait l'incidence d'une arête des cycles $C_1, \ldots, C_i$ à v, contredisant le fait que E_2 soit un 2-couplage. Puisque v est alors extrémité de P, on peut intégrer e_1 ou e_2 au tour partiel $\mathcal{P}_1$, l'une au moins de ces deux arêtes n'étant pas incidente à l'autre extrémité de P. Deux chaînes de $\mathcal{P}_1$ vont être reliées par le biais de l'arête ajoutée, formant une nouvelle chaîne qui alterne encore les arêtes de E_1 et de E_2 ; le résultat est donc de nouveau vrai pour le tour partiel constitué par $\mathcal{P}_1$ à l'itération $i + 1$. Concernant $\mathcal{P}_2$, il est aisé de vérifier que, par construction, il forme bien un tour partiel.

LEMME 1.3.– *[SER 84] Pour toute instance $I = (K_n, d)$ avec n pair, l'inégalité suivante est valable :* $val(\mathcal{P}) \geq \frac{3}{4}\ opt_{maxTSP}(I)$.

Preuve. Soient $I = (K_n, d)$ une instance pour n pair et T^* un tour optimal de I ; dans ce cas, T^* peut se décomposer en deux couplages parfaits (prendre une arête sur deux lors du parcours de T^* pour le premier, les arêtes restantes pour le second), ce dont on déduit :

$$d(E_1) \geq \frac{1}{2} opt_{maxTSP}(I) \qquad [1.9]$$

Par ailleurs, puisqu'un tour optimal est un 2-couplage particulier, sa valeur vérifie également :

$$d(E_2) \geq opt_{maxTSP}(I) \qquad [1.10]$$

Finalement, en regroupant les inégalités [1.9] et [1.10] et comme $val(\mathcal{P}_1)+val(\mathcal{P}_2) = d(E_1) + d(E_2)$, nous obtenons :

$$val(\mathcal{P}) \geq \frac{1}{2}(val(\mathcal{P}_1) + val(\mathcal{P}_2)) \geq \frac{3}{4}\ opt_{maxTSP}(I) \qquad [1.11]$$

∎

Algorithme basé sur le 2-couplage et le couplage :

1) Si n est pair sur $I = (K_n, d)$, alors appliquer l'algorithme meilleur tour partiel sur I (notons $\mathcal{P}$ la solution renvoyée) ;

2) Si n est impair sur $I = (K_n, d)$, alors :

3) Construire K_{n-1} le sous-graphe de K_n induit par $V(K_n) \setminus \{v_1\}$;

4) Pour tout couple $1 < i < j \leq n$, faire :

5) poser $d_{i,j}(e) = d(v_i, v_1)+d(v_1, v_j)$ si $e = [v_i, v_j]$, $d_{i,j}(e) = d(e)$ sinon ;

6) appliquer l'algorithme `meilleur tour partiel` sur l'instance $I_{i,j} = (K_{n-1}, d_{i,j})$ (notons $\mathcal{P}_{i,j}$ la solution renvoyée) ;

7) si $\mathcal{P}_{i,j}$ possède l'arête $[v_i, v_j]$, alors poser $\mathcal{P}_{i,j} \leftarrow (\mathcal{P}_{i,j} \setminus \{[v_i, v_j]\}) \cup \{[v_i, v_1], [v_1, v_j]\}$;

8) Poser $\mathcal{P} = \arg\max_{1<i<j\leq n}\{val(\mathcal{P}_{i,j})\}$;

9) Compléter $\mathcal{P}$ en un tour T de manière arbitraire et renvoyer T.

Nous rappelons que cet algorithme est une version légèrement modifiée de celui présenté dans [SER 84] (une présentation particulièrement claire de l'algorithme original est proposée dans [BAR 02]), essentiellement dans sa façon de traiter le cas n impair. Cette transformation nuit à la complexité en temps de la procédure, qui passe de $O(n^3)$ dans sa version originale à $O(n^5)$ dans le version que nous avons présentée.

THÉORÈME 1.4.– *[SER 84] Pour toute instance $I = (K_n, d)$, l'inégalité suivante est valable :* $val(T) \geq \frac{3}{4} opt_{maxTSP}(I)$.

Preuve. Le tour produit vérifie $val(T) \geq val(\mathcal{P})$, puisque les arêtes sont de coût positif ; si n est pair, le lemme précédent suffit donc à conclure. Supposant maintenant que n est impair sur l'instance $I = (K_n, d)$, de solution optimale T^*, nous considérons l'instance I_{i^*,j^*} où v_{i^*} et v_{j^*} désignent les sommets adjacents à v_1 dans T^* ; on a alors l'inégalité : $opt_{maxTSP}(I_{i^*,j^*}) \geq opt_{maxTSP}(I)$. En effet, le tour T' déterminé par $E(T') = (E(T^*) \setminus \{[v_{i^*}, v_1], [v_1, v_{j^*}]\}) \cup \{[v_{i^*}, v_{j^*}]\}$ est un tour réalisable de I_{i^*,j^*}, de même valeur que T^*. De façon semblable, $\mathcal{P}_{i^*,j^*}$ a même valeur sur I_{i^*,j^*} et sur I (après l'échange de l'arête $[v_{i^*}, v_{j^*}]$ contre la chaîne formée par $[v_{i^*}, v_1]$ et $[v_1, v_{j^*}]$). Couplées au résultat du lemme précédent appliqué à I_{i^*,j^*}, les inégalités que nous venons d'énoncer permettent de conclure :

$$val(\mathcal{P}) \geq val(\mathcal{P}_{i^*,j^*}) \geq \frac{3}{4} opt_{maxTSP}(I_{i^*,j^*}) \geq \frac{3}{4} opt_{maxTSP}(I) \qquad [1.12]$$

∎

La borne produite par cet algorithme (c'est-à-dire l'inégalité [1.12]) était, jusqu'à très récemment, la meilleure connue pour un algorithme déterministe. Depuis peu, cette borne a été améliorée par une déramdomization des algorithmes aléatoires [HAS 00, HAS 02], permettant d'obtenir le rapport standard $\frac{61}{81}$ ($\frac{17}{20}$ dans le cas métrique) pour MAX TSP, voir [CHE 05]. Les algorithmes explicités dans [HAS 00, HAS 02], sont néanmoins plus évolués et garantissent de meilleurs résultats : il s'agit d'algorithmes *aléatoires*, c'est-à-dire qui font des choix dépendants d'une mesure de probabilité. Dans ce cadre, les résultats obtenus ne sont pas édictés pour la valeur du tour, mais pour l'espérance de celle-ci. Ces algorithmes permettent de construire un tour T dont l'espérance de valeur est estimée à au moins $(\frac{25}{33} - \varepsilon) opt_{maxTSP}(I)$ pour tout $\varepsilon > 0$ dans le cas général, à $\frac{7}{8} opt_{max\Delta TSP}(I)$ dans le cas métrique.

1.5. Algorithme approché pour min TSP

Comme cela a été fait dans la section 1.4 pour MAX TSP, nous présentons ici des algorithmes approximant MIN TSP selon les rapports *standard* et *différentiel*.

Puisqu'il s'agit maintenant de la version *minimisation*, la mesure standard est déterminée par l'existence d'un réel $r \geq 1$ vérifiant pour toute instance I, $val(T) \leq r \times opt_{minTSP}(I)$ s'il s'agit du TSP général, $val(T) \leq r \times opt_{min\Delta TSP}(I)$ s'il s'agit de la version métrique (*opt*$_{min\Delta TSP}(I)$ désigne la valeur d'un tour optimal pour MIN METRIC TSP). Pour le rapport différentiel, l'inégalité devient $val(T) \leq r \times opt_{minTSP}(I) + (1-r)wor_{minTSP}(I)$, où cette fois $r \in [0\,;1]$ et $wor_{minTSP}(I)$ est la valeur d'un *plus mauvais tour* , autrement dit : $wor_{minTSP}(I) = opt_{maxTSP}(I)$ (une pire solution pour MIN TSP est solution optimale pour MAX TSP et *vice versa*).

Une propriété naturelle du cas métrique et qui concerne la valeur des solutions optimales est que celle-ci augmente avec l'ordre du graphe. Plus exactement, considérons une instance $I = (K_n, d)$ métrique et soit $I' = (K_{n'}, d)$ sa restriction à un sous-ensemble $V(K_{n'}) \subseteq V(K_n)$; nous avons alors le lemme 1.4.

LEMME 1.4.– *Pour toute instance $I = (K_n, d)$ métrique et pour toute restriction $I' = (K_{n'}, d)$ de I telle que $n' \geq 3$, les valeurs optimales vérifient $opt_{min\Delta TSP}(I') \leq opt_{min\Delta TSP}(I)$.*

Preuve. Soient $I = (K_n, d)$ une instance métrique et $I' = (K_{n'}, d)$ l'une de ses restrictions ; on considère alors un cycle hamiltonien T^* de poids minimal sur I, décrit par la séquence $(v_{i_1}, \ldots, v_{i_n})$. Parcourant cette séquence, on note par v_{j_k} (avec $1 \leq k \leq n'$) le k-ième sommet de $K_{n'}$ rencontré pour la première fois. Il n'est pas difficile de constater que la séquence $(v_{j_1}, \ldots, v_{j_{n'}})$ représente un cycle hamiltonien T' de $K_{n'}$: cette séquence revient à remplacer les chaînes μ_k de T^* d'extrémités v_{j_k} et $v_{j_{k+1}}$ par les arêtes $[v_{j_k}, v_{j_{k+1}}]$; cette opération est souvent appelée « prendre des raccourcis » (voir l'exemple indiqué par la figure 1.10). Puisque d vérifie l'inégalité triangulaire, nous avons pour tout $k = 1, \ldots, n'$ l'inégalité : $d(v_{j_k}, v_{j_{k+1}}) \leq \sum_{e \in \mu_k} d(e)$; en sommant ces inégalités pour tout $k = 1, \ldots, n'$, nous obtenons $opt_{min\Delta TSP}(I') \leq d(T') \leq d(T^*) = opt_{min\Delta TSP}(I)$. ∎

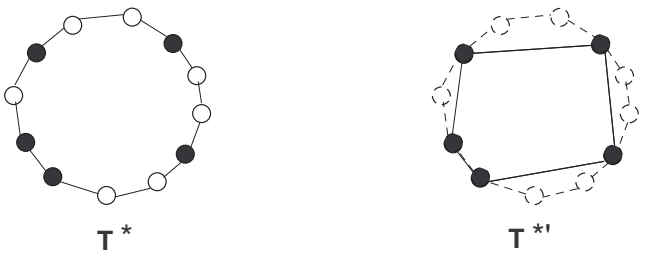

Figure 1.10. *L'opération « prendre des raccourcis »*

Nombreux sont les algorithmes qui approximent MIN METRIC TSP en se basant sur la recherche d'un arbre couvrant de poids minimal et sur le concept de graphe eulérien. Un *graphe eulérien* $G = (V, E)$ est un multigraphe (chaque arête peut apparaître plusieurs fois), connexe, dont tous les sommets sont de degré pair (pour un sommet v, son degré $d_G(v)$ est le nombre d'arêtes de E qui lui sont incidentes). Il n'est pas difficile de prouver que ces graphes sont précisément les graphes qui possèdent un cycle eulérien, c'est-à-dire un cycle passant une et une seule fois par chaque arête, voir [BER 73]. De plus, étant donné un graphe eulérien d'ordre n, on sait trouver un parcours de ce cycle en temps $O(n)$, voir [GON 95].

LEMME 1.5.– *Pour toute instance $I = (K_n, d)$ métrique avec $n \geq 3$ et pour tout multigraphe eulérien $G = (V, E)$ construit sur les sommets de K_n, on peut construire en temps polynomial un cycle hamiltonien T de K_n vérifiant $val(T) \leq d(E)$.*

Preuve. Soient $I = (K_n, d)$ une instance métrique et $G = (V, E)$ un multigraphe eulérien vérifiant $V = V(K_n)$. On note pour un parcours donné de ce cycle eulérien par $(v_{i_1}, \ldots, v_{i_m})$ ($m = |E|$) la séquence des sommets visités. On remarque que chaque sommet v apparaît exactement $\frac{d_G(v)}{2}$ fois dans la séquence et que, puisqu'il s'agit d'un cycle, les sommets v_{i_1} et v_{i_m} coïncident. Pour construire le cycle hamiltonien, on procède de la même façon qu'au cours du lemme 1.4 : partant de v_{i_1} que l'on numérote par 1, on numérote successivement par des entiers naturels de plus en plus grands les sommets de K_n qui sont non encore numérotés et que l'on rencontre pour la première fois lors du parcours de la séquence $(v_{i_1}, \ldots, v_{i_m})$. Le tour T est alors exactement décrit par la séquence des sommets numérotés de 1 à n. Par construction, tous les sommets de T ne sont visités qu'une seule fois ; de plus, puisque G est connexe, on est assuré qu'aucun sommet n'est oublié : T est bien un cycle hamiltonien. Supposons que celui-ci soit décrit par la séquence $(v_{j_1}, \ldots, v_{j_n})$ et notons par μ_k pour $k = 1, \ldots, n$ l'ensemble des arêtes de E rencontrées lors du parcours de la séquence $(v_{i_1}, \ldots, v_{i_m})$ qui commence au sommet numéroté k et s'achève au sommet d'ordre $k + 1$ (ces indices sont considérés *modulo* n). Puisque la séquence $(v_{i_1}, \ldots, v_{i_m})$ décrit un cycle eulérien, les chaînes μ_k forment une partition de E. De plus, puisque d vérifie l'inégalité triangulaire, nous avons pour tout $k = 1, \ldots, n$ l'inégalité : $d(v_{j_k}, v_{j_{k+1}}) \leq \sum_{e \in E(\mu_k)} d(e)$. En sommant ces inégalités pour tout $k = 1, \ldots, n$, nous obtenons le résultat. Enfin, nous remarquons que cette procédure est réalisable en un temps $O(n)$. ∎

En s'aidant de ce lemme, il est aisé de bâtir un algorithme fondé sur la recherche d'un arbre couvrant de poids minimal. Etant donné un graphe connexe $G = (V, E)$, un *arbre couvrant* G est un sous-ensemble d'arêtes $E' \subseteq E$ tel que le graphe partiel $G' = (V, E')$ est connexe et sans cycle. Nous indiquons au lecteur qu'il existe de nombreuses propriétés et caractérisations des arbres, voir [BER 73]. Lorsque chaque arête e du graphe $G = (V, E)$ a un poids $d(e) \geq 0$, la détermination d'un arbre couvrant E' dont la somme des poids des arêtes est minimale parmi les arbres couvrants de G peut

se faire en complexité $O(n^2)$, voir [GON 95]. L'ensemble E' est alors appelé *arbre couvrant de poids minimal*.

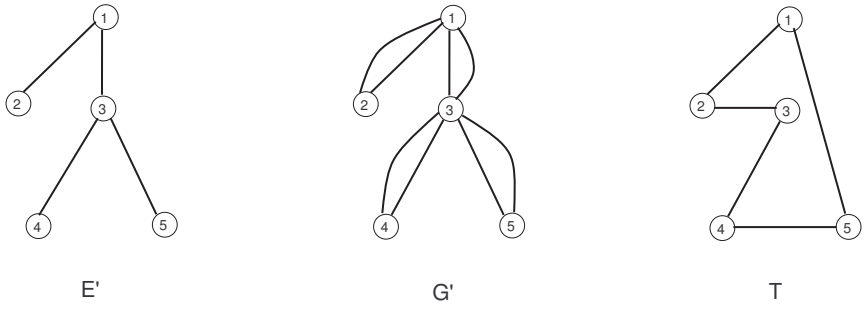

Figure 1.11. *La solution T*

Algorithme basé sur l'arbre couvrant :

1) Construire un arbre E' couvrant K_n de poids minimal ;

2) Construire le multigraphe G' sur les sommets de K_n en partant du graphe partiel induit par E' et en dupliquant chaque arête de E' ;

3) Appliquer sur G' le lemme précédent et renvoyer T.

D'après les explications précédentes, la complexité en temps de cet algorithme vaut $O(n^2)$ et est imputable à la construction établie par l'étape 1 ; vérifions maintenant que cet algorithme produit bien un cycle hamiltonien T, ce qui résume à démontrer que G' est eulérien. Or, d'une part, G' est connexe car il contient l'arbre E' et, d'autre part, le degré dans G' de chaque sommet est pair puisqu'il est le double du degré dans le graphe partiel engendré par l'arbre E' (la figure 1.11 illustre le fonctionnement de cet algorithme) !

Considérons à présent la performance, en termes de qualité de la solution fournie, de l'algorithme ; pour ce, nous évoquons le théorème 1.5, qui n'a pas de référence exacte ; disons qu'il appartient au « folklore » de la discipline.

THÉORÈME 1.5.– *Pour toute instance $I = (K_n, d)$ métrique, l'inégalité suivante est valable : $val(T) \leq 2\ opt_{min\Delta TSP}(I)$.*

Preuve. Soient $I = (K_n, d)$ une instance métrique et T^* une solution optimale de valeur $opt_{min\Delta TSP}(I)$ du voyageur de commerce ; la suppression d'une arête e quelconque de T^* forme un arbre couvrant de K_n et puisque $d(e) \geq 0$, nous déduisons

$d(T^*) - d(e) \leq opt_{min\Delta TSP}(I)$. D'un autre côté, la valeur de E' est celle d'un arbre couvrant de poids minimal. Ainsi, nous avons :

$$d(E') =\leq opt_{min\Delta TSP}(I) \quad [1.13]$$

Finalement, puisque le poids des arêtes de G' vaut $2d(E')$, le lemme précédent appliqué à G' nous permet de conclure. ∎

1.5.1. *Algorithme basé sur l'arbre couvrant et le couplage*

L'algorithme proposé dans [CHR 76] est un raffinement de la méthode précédente ; il est donc construit à partir d'un arbre couvrant de poids minimal, mais également à partir d'un couplage parfait de poids minimal. Un *couplage* dans un graphe $G = (V, E)$ est un ensemble d'arêtes $E' \subseteq E$ deux à deux non adjacentes. Ce couplage est dit *parfait* lorsque, dans G', il n'existe pas de sommet isolé (chaque sommet est l'extrémité d'une arête). Dans ce cas, il est nécessaire que l'ordre du graphe n soit pair et nous avons la relation $|E'| = \frac{n}{2}$. Un couplage dans un graphe G dont les arêtes sont pondérées par une fonction distance d est dit *parfait de poids minimal* lorsque la somme des poids de ses arêtes est minimale parmi les couplages parfaits de G. La détermination d'un couplage parfait de poids minimal est de complexité polynomiale et peut se faire en $O(n^3)$, voir [GON 95, PAP 98]. Notons cependant que la détermination d'un couplage de poids minimal parmi les couplages maximaux vis-à-vis de l'inclusion (l'ajout d'une arête quelconque au couplage ne forme plus un couplage) est un problème **NP**-difficile, et ce même lorsque les poids sont tous égaux, voir [GAR 79].

Algorithme basé sur l'arbre couvrant et le couplage :

1) Construire un arbre E' couvrant K_n de poids minimal ;

2) Construire la restriction $I' = (K_{n'}, d')$ de I aux sommets V' dont le degré est impair dans le graphe partiel G' engendré par les arêtes de l'arbre E' ;

3) Construire un couplage parfait E'' de poids minimal sur I' ;

4) Construire le multigraphe G'' dont les sommets sont $V(K_n)$ et les arêtes E' et E'' (les arêtes communes à E' et E'' sont dupliquées) ;

5) Appliquer sur G'', le lemme précédent et renvoyer T ;

La complexité de cet algorithme est principalement déterminée par l'itération 3, qui consomme un temps $O(n^3)$, tandis que sa validité est conditionnée par le fait que les itérations 3 et 5 soient possibles à effectuer ; or, puisque le graphe de l'instance I' est complet, un couplage parfait existe si et seulement si $|V'|$ est pair : il s'agit donc de prouver que le nombre des sommets de degré impair dans G' est pair. En réalité, cela

est vrai pour tout graphe $G = (V, E)$ et provient de l'égalité $\sum_{v \in V} d_G(v) = 2|E|$; ainsi, l'itération 3 est bien valide. Concernant l'itération 5, il suffit de vérifier que G'' est bien un multigraphe eulérien. Or, d'une part, G'' est connexe puisqu'il contient l'arbre couvrant E' et, d'autre part, tous ses sommets sont de degré pair puisque seuls les sommets de degré impair dans G' voient leur degré augmenté d'une unité dans G''. Une illustration de cet algorithme est proposé dans la figure 1.12.

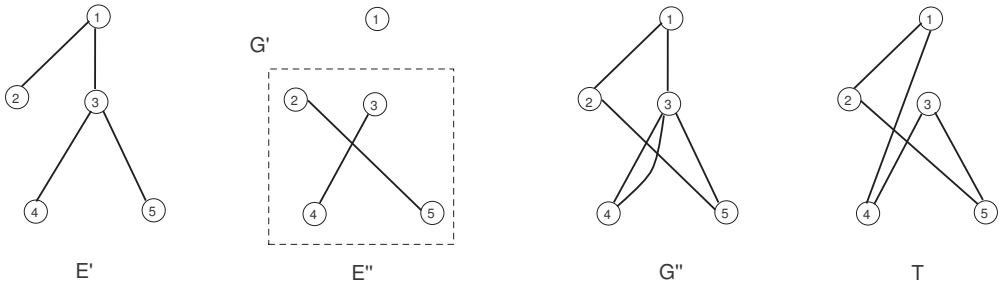

Figure 1.12. *La construction de la solution T*

THÉORÈME 1.6.– *[CHR 76] Pour toute instance $I = (K_n, d)$ métrique, l'inégalité suivante est valable : $val(T) \leq \frac{3}{2}\, opt_{min\Delta TSP}(I)$.*

Preuve. Soit $I = (K_n, d)$ une instance métrique ; en appliquant le lemme 1.4 sur l'instance I', nous avons :

$$d(T_1^*) \leq opt_{min\Delta TSP}(I) \qquad [1.14]$$

où T_1^* est une solution optimale sur l'instance I'. Par ailleurs, les arêtes du tour T_1^* peuvent s'écrire comme l'union de deux couplages parfaits E_1 et E_2 qui sont chacun de valeur supérieure ou égale à la valeur du couplage E'' de poids minimal sur I'. Ainsi, nous obtenons :

$$2d(E'') = 2d'(E'') \leq d'(E_1) + d'(E_2) \leq opt_{min\Delta TSP}(I) \qquad [1.15]$$

En ajoutant cette inégalité à l'inégalité [1.13] et en appliquant le lemme 1.5 sur G'', nous déduisons le résultat annoncé, c'est-à-dire :

$$val(T) \leq d(E') + d(E'') \leq \frac{3}{2} opt_{min\Delta TSP}(I) \qquad [1.16]$$

Il est intéressant de noter que cet algorithme, conçu en 1976, garantit encore à l'heure actuelle le meilleur rapport de performance standard connu pour la version métrique du problème de voyageur de commerce.

1.5.2. *Algorithme de recherche locale*

Nous nous intéressons à présent à une procédure largement utilisée dans la pratique : *l'amélioration locale*. Divers algorithmes de ce type, ainsi que l'analyse de leurs performances selon les mesures standard et différentielle, sont discutés dans [AAR 97, MON 03, PAP 98]. Grossièrement, un algorithme d'amélioration locale est construit autour de la notion de voisinage et d'une procédure d'amélioration dans ce voisinage, voir le chapitre 4 du volume 2 écrit par E. Angel, P. Christopoulos, V. Zissimopoulos. Concrètement, étant donné un tour T, un *voisinage* de T est un ensemble de tours relativement proches de T ; la *procédure d'amélioration* consiste alors à renvoyer un tour T' de ce voisinage de valeur strictement meilleure si un tel tour existe, à retourner T sinon. Par exemple, dans [LIN 73], les auteurs ont proposé des voisinages de T constitués des tours T' dont le nombre d'arêtes qui diffèrent de T est majoré par k. Ainsi, étant donnés un type de voisinage et une procédure d'amélioration, l'algorithme se déroule comme suit : partant d'un cycle hamiltonien quelconque T, on cherche par le biais de la procédure d'amélioration un tour T' de meilleure valeur dans le voisinage de T ; tant qu'un tel tour existe, on remplace T par T' et on réitère la procédure ; à la fin de l'algorithme, le dernier tour T trouvé ne peut être amélioré par l'échange avec un tour voisin et dans ce cas, T est qualifié d'*optimum local* relativement au voisinage considéré.

Précisons que cet algorithme s'arrêtera toujours puisque nous étudions des problèmes d'optimisation combinatoire qui ont par définition un nombre fini de solutions réalisables (ou encore, un nombre fini de valeurs possibles) ; cependant, la complexité en temps de tels algorithmes dépendra conjointement de la procédure d'amélioration et du nombre d'itérations réalisées avant d'atteindre un optimum local. Par exemple, pour un cycle hamiltonien T quelconque, si la taille du voisinage de T est bornée par un polynôme $p(n)$, alors la procédure d'amélioration est de complexité polynomiale (au pire des cas, on teste la valeur de tous ses voisins et l'on en déduit celui de meilleure valeur) ; cependant, cela ne garantira pas que l'algorithme soit de complexité polynomiale comme nous le verrons plus tard. D'ailleurs, l'existence de voisinage de taille exponentielle qui est pourtant examinable en temps polynomial a récemment été révélée dans [GUT 99].

Du point de vue de la performance de tels algorithmes, plusieurs résultats négatifs ont été obtenus dans [PAP 77, PAP 78] ; il a notamment été démontré que, sous l'hypothèse **P≠NP**, les algorithmes d'amélioration locale qui utilisent une procédure d'amélioration polynomiale, mais éventuellement un nombre de fois exponentiel, ne vérifieront jamais pour toute instance $I = (K_n, d)$ l'inégalité $d(T) \leq r \times opt_{minTSP}(I)$, où $r \geq 1$ est un réel quelconque (précisons que ce résultat est valable pour le cas général, et donc lorsque la fonction distance ne vérifie pas nécessairement l'inégalité triangulaire). Nous allons voir qu'en utilisant un autre type de mesure, des résultats positifs sur ces algorithmes sont toutefois envisageables.

L'algorithme d'amélioration locale que nous allons étudier fait appel à un type de voisinage particulier décrit dans [LIN 73], appelé *2-échange*. Pour ce voisinage, un tour T' est voisin d'un tour T si T' résulte de T après l'échange de 2 arêtes e et e' de T par 2 nouvelles arêtes ; une illustration de ce procédé est proposée par la figure 1.13. Précisons que lorsque les arêtes e et e' sont fixées, il n'existe qu'une seule manière de former T' par un 2-échange. Originellement conçu dans [CRO 58], l'algorithme que

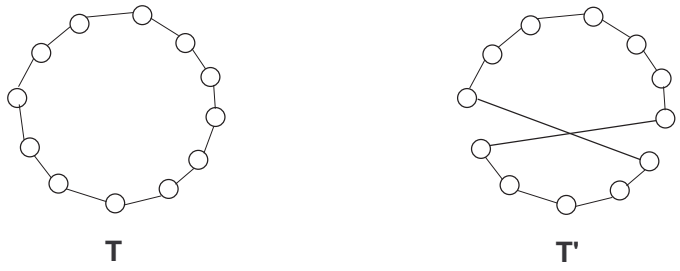

Figure 1.13. *Le 2-échange*

nous étudions est souvent désigné par `2-opt` dans la littérature ; cet algorithme a été très fréquemment revisité au cours de diverses tentatives de résolution de MIN TSP, voir [JOH 97, LIN 73].

Algorithme 2-opt :

1) Trouver un cycle hamiltonien T quelconque ;

2) Poser $\lambda = -1$;

3) Tant que $\lambda < 0$, faire :

4) Chercher 2 arêtes $e = [v_i, v_j]$ et $e' = [v_{i'}, v_{j'}]$ du tour qui vérifient $\lambda = d(e) + d(e') - (d(v_i, v_{i'}) + d(v_j, v_{j'})) < 0$, poser $\lambda = 0$ s'il n'y en a pas ;

5) Si $\lambda < 0$, alors $E(T) \leftarrow (E(T) \setminus \{e, e'\}) \cup \{[v_i, v_{i'}], [v_j, v_{j'}]\}$;

6) Renvoyer T.

Nous commençons par quelques mots quant à la complexité en temps de `2-opt` : tout d'abord, sur une instance $I = (K_n, d)$, la taille du voisinage d'un tour T quelconque est majoré par n^2 (il y a au plus n choix possibles pour e et autant pour e') ; ainsi, la procédure d'amélioration (étapes 3.1 et 3.2) nécessite un temps $O(n^2)$ et par conséquent, la complexité en temps de cet algorithme dépend essentiellement du nombre d'itérations qu'il faudra effectuer avant d'obtenir un optimum local. Malheureusement, il est prouvé dans [FIS 95] qu'étant donné un cycle hamiltonien initial T de K_n, décider seulement s'il est possible d'atteindre en temps polynomial un optimum local à partir de T par l'algorithme `2-opt` est en lui-même un problème **NP**-difficile ; en d'autres termes, cet algorithme n'est pas, sous l'hypothèse **P=NP**, de complexité polynomiale. Proposons une configuration dans laquelle cet algorithme devient polynomial : lorsque la difference entre les valeurs des solutions extrêmes est borné par un polynôme p (c'est-à-dire $|wor_{minTSP}(I) - opt_{minTSP}(I)| \leq p(n)$, où $I = (K_n, d)$). En effet, le tour initial (correspondant à l'étape 1) est au pire des cas de valeur $wor_{minTSP}(I)$ et à chaque itération de la boucle de l'étape 3, la valeur du cycle courant augmente d'au moins une unité puisque les valeurs sont entières ; ainsi, en au plus $p(n)$ itérations, nous obtenons un optimum local. En conclusion, la complexité en temps de la recherche d'un optimum local est au pire de $O(n^2 \times p(n))$. Un exemple concret de réalisation de cette situation est le cas $d_{\max} \leq n^k$ pour un certain entier k ; précisons cependant que d'autres cas de polynomialité sont présentés dans [MON 02c].

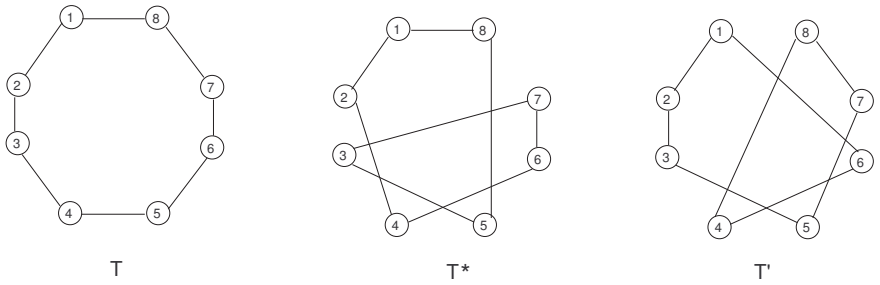

Figure 1.14. *Construction de T' à partir des solutions T et T^**

THÉORÈME 1.7.– *[MON 02c] Pour toute instance $I = (K_n, d)$, l'inégalité suivante est valable : $val(T) \leq \frac{1}{2} opt_{minTSP}(I) + \frac{1}{2} wor_{minTSP}(I)$.*

Preuve. Soit $I = (K_n, d)$ une instance du voyageur de commerce ; nous utilisons ici la notation fonctionnelle explicitée dans la section *propriétés élémentaires* pour

décrire les cycles hamiltoniens. Ainsi, soit f^* une permutation acyclique de $V(K_n)$ vérifiant la propriété [1.1] et décrivant un tour optimal T^* de valeur $opt_{minTSP}(I)$. Cette valeur s'écrit : $d(T^*) = \sum_{v \in V(K_n)} d(v, f^*(v))$. De même, notons par f une permutation correspondant à la solution T produite par l'algorithme. On a alors : $val(T) = \sum_{v \in V(K_n)} d(v, f(v))$. L'optimalité locale de T se traduit par le fait que l'échange des arêtes $[v, f(v)]$ et $[f^*(v), f \circ f^*(v)]$ de $E(T)$ par les 2 arêtes $[v, f^*(v)]$ et $[f(v), f \circ f^*(v)]$ n'augmente pas la valeur du tour. Autrement dit, nous avons pour tout $v \in V(K_n)$:

$$d(v, f(v)) + d(f^*(v), f \circ f^*(v)) \leq d(v, f^*(v)) + d(f(v), f \circ f^*(v)) \quad [1.17]$$

De plus, la fonction f^* étant une permutation, sa valeur peut s'exprimer comme suit : $val(T) = \sum_{v \in V(K_n)} d(f^*(v), f \circ f^*(v))$. De même, $\sum_{v \in V(K_n)} d(f(v), f \circ f^*(v)) = \sum_{v \in V(K_n)} d(v, f \circ f^* \circ f^{-1}(v))$ où f^{-1} désigne la permutation réciproque de f. Il est aisé de vérifier que $f \circ f^* \circ f^{-1}$ est une permutation acyclique (c'est-à-dire, satisfaisant la propriété [1.1]) et donc, correspond à un tour T' (voir la figure 1.14) de valeur $d(T') = \sum_{v \in V(K_n)} d(f(v), f \circ f^*(v))$. En sommant les inégalités [1.17] pour tout $v \in V(K_n)$ et en tenant compte des trois inégalités que nous venons d'établir, nous obtenons :

$$2val(T) \leq d(T^*) + d(T') \leq opt_{minTSP}(I) + wor_{minTSP}(I) \quad [1.18]$$

∎

Notons que l'inégalité résultant du théorème 1.7 est la plus fine possible écrite sous cette forme. Nous laissons au lecteur le soin de prouver cette assertion.

1.6. Algorithmes constructifs

Les *algorithmes constructifs* sont proches des algorithmes gloutons, dans le sens où ils consistent à faire un choix à chaque itération sans jamais remettre ce choix en cause. Ces algorithmes sont très simples à implémenter et, en général, rapides à exécuter, comme cela est montré dans [JOH 02]. Dans notre contexte, cela se traduit par l'introduction d'un sommet à chaque itération en vue d'établir un parcours des sommets. Le plus connu d'entre eux est certainement celui appelé du *plus proche voisin* et c'est celui-ci que nous allons étudier maintenant. Précisons cependant qu'il en existe beaucoup d'autres, explicités dans [LAW 85, ROS 77] ; parmi ceux-ci, citons l'algorithme de la *plus proche insertion*, voir [KAR 64] que nous étudierons ensuite, ou encore l'algorithme de la *meilleure insertion*, voir [NIC 67].

1.6.1. *Algorithme du plus proche voisin*

L'algorithme que nous étudions consiste à partir d'un sommet arbitraire et à construire un parcours de T en visitant itérativement un sommet non encore visité qui est le plus proche (ou le plus éloigné suivant que l'on étudie MIN TSP ou MAX TSP) du dernier sommet visité. Cet algorithme a été décrit pour la première fois dans [GAV 65] et l'analyse de ses performances a été effectuée dans [ROS 77] pour la version MIN METRIC TSP, dans [FIS 79] pour la version MAX TSP. Dans le premier cas, les auteurs ont démontré l'inégalité $val(T) \leq \frac{1}{2}(\lceil log_2 n \rceil + 1) opt_{min\Delta TSP}(I)$ et ils ont exhibé des instances $I_n = (K_n, d)$ où $val(T_n)$ est équivalent à $\frac{1}{3} log_2 n \times opt_{min\Delta TSP}(I_n)$ lorsque n tend vers l'infini ; ainsi, asymptotiquement, cette inégalité est la meilleure qui soit. Notons également que d'autres familles d'instances, plus simples à construire et indiquant toujours que la valeur du tour trouvé est asymptotiquement égale à $O(log_2 n \times opt_{min\Delta TSP}(I))$, sont présentées dans [HUR 04, JOH 85]. Concernant la version maximisation, les résultats sont plus optimistes puisque dans [FIS 79], les auteurs ont démontré que l'inégalité $val(T) \geq \frac{1}{2} opt_{maxTSP}(I)$ est valide pour toute instance $I = (K_n, d)$ de MAX TSP. Nous allons raffiner ici ces deux inégalités en démontrant que lorsque la fonction distance prend ses valeurs dans l'ensemble $\{a, a+1, \ldots, b\}$, nous avons les inégalités $val(T) \geq \frac{t+1}{2t} opt_{maxTSP}(I)$ pour MAX TSP et $val(T) \leq \frac{t+1}{2} opt_{minTSP}(I)$ pour MIN TSP, où dans les deux cas $t = \frac{b}{a}$. La démonstration de ces analyses a été établie dans [MON 02a].

Afin d'expliciter formellement cet algorithme, nous utilisons la notation fonctionnelle décrite dans la section *Propriétés élémentaires* pour caractériser les cycles hamiltoniens. De plus, cet algorithme dépendra d'un paramètre appelé obj indiquant si l'on étudie la version maximisation ou minimisation.

Algorithme plus proche voisin pour obj TSP :

1) Prendre arbitrairement $v \in V(K_n)$;
2) Poser $V' = \{v\}$ et $z = v$;
3) Tant que $V' \neq V(K_n)$, faire :
4) Trouver $v \notin V'$ tel que $d(z, v) = obj\{d(z, v') : v' \notin V'\}$;
5) Poser $f(z) = v$, $z = v$ et $V' \longleftarrow V' \cup \{v\}$;
6) Poser $f(z) = v$ et renvoyer f.

Si durant l'étape 4, il existe plusieurs sommets v vérifiant la propriété, alors l'algorithme prend le premier sommet rencontré qui le satisfait. On peut aisément voir que f est une permutation de $V(K_n)$ pour laquelle la propriété [1.1] est valide, et donc que f correspond à un tour T de $I = (K_n, d)$. En termes de complexité, cet algorithme consomme un temps $O(n^2)$.

1.6.1.1. *Le cas général*

Pour l'analyse de performance qui suit, nous supposons d'une part que a et $t \times a$ sont entiers avec $t > 1$, d'autre part que les distances vérifient $d(e) \in \{a, \ldots, ta\}$.

THÉORÈME 1.8.– *[FIS 79, MON 02a] Lorsque $obj = max$, pour toute instance $I = (K_n, d)$, on a $val(T) \geq \frac{t+1}{2t} opt_{maxTSP}(I)$.*

Preuve. Soit $I = (K_n, d)$ une instance de MAX TSP vérifiant $\forall e \in E(K_n), a \leq d(e) \leq ta$; si f^* désigne une permutation de $V(K_n)$ correspondant à une solution optimale T^*, nous avons : $opt_{maxTSP}(I) = \sum_{v \in V(K_n)} d(v, f^*(v))$. De plus, comme T^* possède exactement n arêtes et que par définition $a \leq d(e) \leq ta$, on déduit en sommant ces inégalités sur les arêtes de T^* : $an \leq opt_{maxTSP}(I) \leq nta$. Par ailleurs, concernant la permutation f correspondant au tour T, nous avons : $val(T) = \sum_{v \in V(K_n)} d(v, f(v)) = \sum_{v \in V(K_n)} d(f^*(v), f \circ f^*(v))$.

Afin de clarifier la preuve, nous identifions un sommet v à l'itération où il a été incorporé à V' durant l'étape 5. Nous partitionnons les sommets $V(K_n)$ en deux sous-ensembles $V_1 = \{v \in V(K_n) : \ d(v, f(v)) < d(v, f^*(v))\}$ et $V_2 = V(K_n) \setminus V_1$; en quelque sorte, V_1 (resp., V_2) correspond aux itérations où l'algorithme s'est (resp. ne s'est pas) trompé par rapport au choix de l'optimum. Ainsi, au départ, l'algorithme ne se trompe pas, c'est-à-dire que le sommet v choisi à l'étape 1 vérifie $v \in V_2$. De plus, si l'algorithme ne se trompe jamais (c'est-à-dire $V_1 = \emptyset$), alors il produit une solution optimale. Enfin, à chaque fois que l'algorithme se trompe, on peut faire correspondre une itération où il ne se trompe pas : $\forall v \in V_1, \ d(f^*(v), f \circ f^*(v)) \geq d(v, f^*(v))$. En effet, si l'algorithme n'a pas choisi $f^*(v)$ à l'itération v, c'est que ce sommet a été incorporé à la solution lors d'une itération antérieure (c'est-à-dire $f^*(v) \in V'$ à l'itération v) ; de fait, v était disponible lors de la considération de $f^*(v)$ (c'est-à-dire $v \notin V'$ à l'itération $f^*(v)$) et l'algorithme a pu faire alors un choix au moins aussi bon que celui de l'optimum. En rapprochant ces différentes inégalités, nous démontrons le résultat souhaité. ∎

THÉORÈME 1.9.– *[MON 02a] Lorsque $obj = min$, pour toute instance $I = (K_n, d)$, on a $val(T) \leq \frac{t+1}{2} opt_{minTSP}(I)$.*

Preuve. Soit $I = (K_n, d)$ une instance de MIN TSP, nous construisons l'instance $I' = (K_n, d')$ de MAX TSP en posant $d'(e) = d_{max} + d_{min} - d(e)$, comme cela a été fait au cours de la section *Propriété élémentaires*. La clé du raisonnement consiste à prouver que la solution f renvoyée par plus proche voisin pour MIN TSP sur I est la même que la solution f' renvoyée par `plus proche voisin` pour MAX TSP sur I'. Ainsi, si T désigne le tour correspondant à f et f', nous déduirons : $val(T) = n(d_{max} + d_{min}) - d'(T)$.

Concernant respectivement les solutions optimales de MIN TSP sur I et de MAX TSP sur I', l'égalité [1.4] nous indique qu'elles coincïdent. Ainsi, nous avons la relation $opt_{minTSP}(I) = n(d_{max} + d_{min}) - opt_{maxTSP}(I')$. De plus, puisque l'on a $opt_{minTSP}(I) \geq n \times d_{min}$, nous déduisons $opt_{maxTSP}(I') \leq t \times opt_{minTSP}(I)$. Enfin, il est clair que l'instance I' vérifie $\forall e \in E(K_n), a \leq d'(e) \leq ta$ si I satisfait $a \leq d'(e) \leq ta$; on peut donc appliquer le théorème 1.8 pour l'instance I' et en utilisant les égalités établies, nous obtenons bien le résultat attendu. Nous laissons au lecteur le soin de prouver que les deux inégalités énoncées par ces théorèmes sont également les meilleures possibles écrites sous cette forme. ∎

1.6.1.2. *Le cas métrique*

Dans le théorème 1.9, nous avons obtenu le ratio $\frac{d_{\max}}{2d_{\min}} + \frac{1}{2}$ lorsque $obj = min$, ce qui peut être aussi mauvais que $O(2^n)$. Nous allons démontrer maintenant que le rapport est d'ordre logarithmique lorsque la fonction d vérifie l'inégalité triangulaire.

THÉORÈME 1.10.– *[ROS 77] Lorsque $obj = min$, pour toute instance métrique $I = (K_n, d)$, on a $val(T) \leq \frac{1}{2}(\lceil \log_2 n \rceil + 1) opt_{min\Delta TSP}(I)$.*

Preuve. Soient $I = (K_n, d)$ une instance métrique de MIN TSP, $T = \{[v_i, f(v_i)] : i = 1, \ldots, n\}$ le tour produit par l'algorithme et T^* un tour optimal ; notant $d_i = d(v_i, f(v_i))$, nous affirmons que pour tout $i, j \in \{1, \ldots, n\}$, $i \neq j$, les distances vérifient :

$$d(v_i, v_j) \geq \min\{d_i, d_j\} \qquad [1.19]$$

En effet, par construction de l'algorithme, nous avons $d(v_i, v_j) \geq d_i$ dès lors que le sommet v_i est considéré par l'algorithme avant le sommet v_j. Afin de simplifier les notations, considérons une permutation $\ell_1, \ldots, \ell_n$ sur $\{1, \ldots, n\}$ vérifiant $d_{\ell_1} \geq \ldots \geq d_{\ell_n}$ (autrement dit, une permutation qui permet d'ordonner les sommets $f^1(v), \ldots, f^n(v)$ en fonction de la distance de l'arête $[f^{i-1}(v), f^i(v)]$ prise par la solution T) ; nous démontrons que pour tout $k = 1, \ldots, n$, la valeur optimale vérifie :

$$opt_{min\Delta TSP}(I) \geq 2 \sum_{i=k+1}^{\min\{2k,n\}} d_{\ell_i} \qquad [1.20]$$

Soit $k \in \{1, \ldots, n\}$, nous considérons T_k^* le tour résultant de la trace de T^* sur le sous-graphe partiel induit par les sommets $\{f^{\ell_1 - 1}(v), \ldots, f^{\ell_p - 1}(v)\}$, où $p =$

$\min\{2k, n\}$. La trace de T^* sur ce sous-graphe partiel consiste à parcourir ces sommets conformément à leur ordre de parcours dans T^* ; la figure 1.15 propose une illustration de la construction de T_k^* pour $k = 3$ et $p = 2k$. Puisque d satisfait l'inégalité triangulaire, nous avons immédiatement :

$$opt_{min\Delta TSP}(I) \geq d(T_k^*) \qquad [1.21]$$

En utilisant l'inégalité [1.19], nous déduisons également :

$$d(T_k^*) \geq \sum_{[v_{\ell_i}, v_{\ell_j}] \in T_k^*} \min\{d_{\ell_i}, d_{\ell_j}\} = \sum_{i=1}^{p} \alpha_{\ell_i} d_{\ell_i} \qquad [1.22]$$

où α_{ℓ_i} est le nombre de fois que $\min\{d_{\ell_i}, d_{\ell_j}\} = d_{\ell_i}$ dans la somme précédente, c'est-à-dire le nombre de sommets $f^{\ell_j - 1}(v)$ adjacents à $f^{\ell_i - 1}(v)$ dans T_k^* pour lesquels $d_{\ell_j} \leqslant d_{\ell_i}$. On a $0 \leq \alpha_{\ell_i} \leq 2$ et $\sum_{i=1}^{p} \alpha_{\ell_i} = p$. Par exemple, si dans la figure 1.15 nous supposons $\ell_i = i$, nous avons $\alpha_1 = \alpha_2 = 0$, $\alpha_3 = \alpha_6 = 2$ et $\alpha_4 = \alpha_5 = 1$.

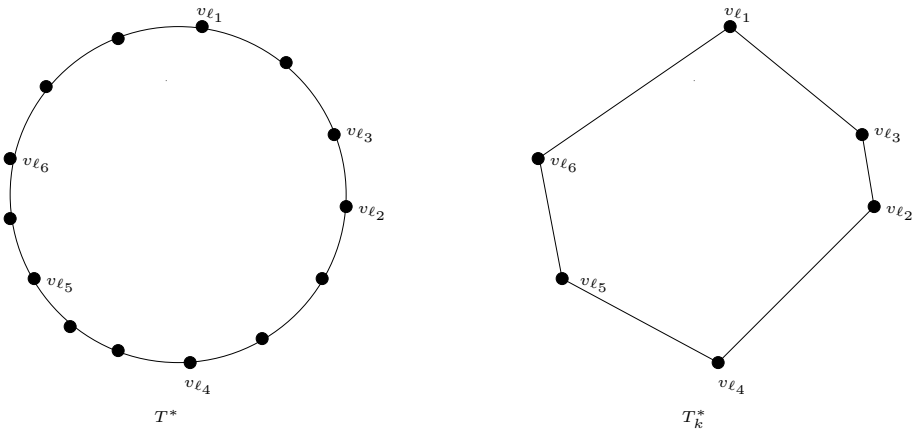

Figure 1.15. *Le tour T_k^* pour $k = 3$*

Puisque $p \leq 2k$ et que les d_{ℓ_i} sont triés par ordre décroissant, une pire configuration de l'inégalité [1.22] (la plus petite borne inférieure que l'on puisse en déduire pour la valeur de T_k^*) est donnée lorsque $\alpha_1 = \ldots = \alpha_k = 0$ et $\alpha_{k+1} = \ldots = \alpha_p = 2$, c'est-à-dire : $d(T_k^*) \geqslant 2\sum_{i=k+1}^{p} d_{\ell_i}$. Selon l'inégalité [1.21], l'inégalité [1.20] est ainsi démontrée.

En sommant maintenant l'inégalité [1.20] pour $k = 2^j$ lorsque j varie de 0 à $\lceil log_2 n \rceil - 1$, nous obtenons :

$$\lceil log_2 n \rceil \times opt_{min\Delta TSP}(I) \geq 2 \sum_{j=0}^{\lceil log_2 n \rceil - 1} \sum_{i=2^j+1}^{\min\{2^{j+1},n\}} d_{\ell_i} = 2 \sum_{i=2}^{n} d_{\ell_i} \quad [1.23]$$

Par ailleurs, nous avons :

$$d_{\ell_1} \leq \frac{1}{2} opt_{min\Delta TSP}(I) \quad [1.24]$$

En effet, T^* étant un tour, il peut se décomposer en 2 chaînes E_1 et E_2 qui relient respectivement $f^{\ell_1 - 1}(v)$ à $f^{\ell_1}(v)$ et $f^{\ell_1}(v)$ à $f^{\ell_1 - 1}(v)$. Or, puisque la distance d vérifie l'inégalité triangulaire, la valeur de ces chaînes vérifie $d(E_j) \geq d_{\ell_1}$ pour $j = 1, 2$ et ainsi, $opt_{min\Delta TSP}(I) = d(E_1) + d(E_2) \geq 2d_{\ell_1}$. Cette dernière inégalité à présent établie et sachant $val(T) = \sum_{i=1}^{n} d_{\ell_i}$, il n'y a plus qu'à la sommer à la relation [1.23] pour conclure. ∎

Explicitons maintenant une famille d'instances $I_k = (K_n, d)$, $k \geq 1$ pour laquelle le ratio de performance de l'algorithme `plus proche voisin` est proportionnel à $O(\log_2 n)$. L'instance I_n est définie à partir d'un graphe connexe G_k, que l'on complète de sorte que $d(v_i, v_j)$ soit la distance d'un plus court chemin de v_i à v_j dans le graphe initial G_k. La suite des graphes G_k est définie formellement de la manière suivante :

– G_1 est un triangle sur les sommets l_1, r_1, m_1 ;

– étant donné G_{k-1}, le graphe G_k est construit à partir de 2 copies G^1_{k-1} et G^2_{k-1} de G_{k-1} possédant chacune 3 sommets particuliers l^1_{k-1}, r^1_{k-1}, m^1_{k-1} et l^2_{k-1}, r^2_{k-1}, m^2_{k-1} ; pour obtenir G_k, on ajoute un nouveau sommet m_k, on renomme l^1_{k-1} en l_k et r^2_{k-1} en r_k, puis on relie deux à deux les sommets r^1_{k-1}, l^2_{k-1} et m_k. Ainsi, G_k consiste à relier les graphes G^1_{k-1} et G^2_{k-1} par l'intermédiaire d'un triangle formé de l'extrémité droite de G^1_{k-1}, de l'extrémité gauche de G^2_{k-1} et d'un nouveau sommet m_k.

La figure 1.16 illustre la construction de G_3 à partir de G^1_2 et G^2_2. Le graphe G_k se présente comme une chaîne de $2^k - 1$ triangles et il y a respectivement 2^k sommets et $2^k - 1$ sommets « en bas » et « en haut » des triangles ; ainsi, le sommet m_k se retrouve à égale distance, précisément à une distance de 2^{k-1}, des sommets l_k et r_k.

LEMME 1.6.– *L'algorithme plus proche voisin appliqué à l'instance I_k produit, avant la dernière itération, une chaîne hamiltonienne T_k de l_k à m_k de valeur $val(T_k) = (k+3)2^{k-1} - 2$.*

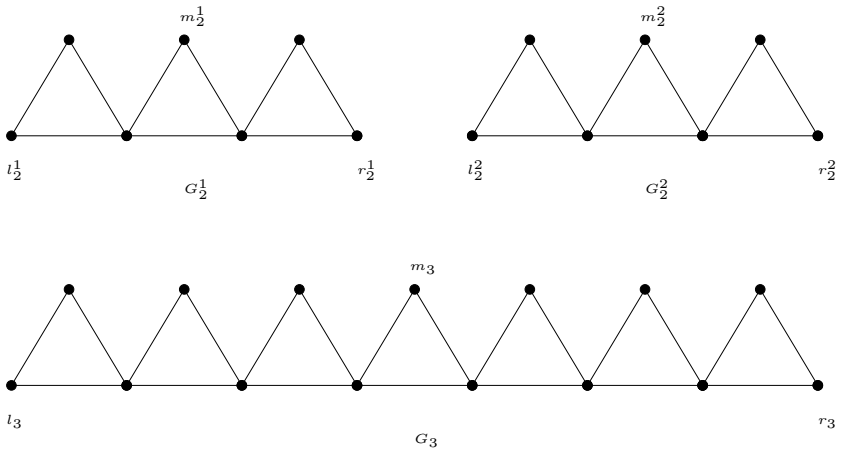

Figure 1.16. *La construction de G_3.*

Preuve. Démontrons ce résultat par induction. Pour I_1, la solution T_1 part de l_1, va visiter r_1 et termine son parcours en m_1 ; c'est une chaîne hamiltonienne de valeur $val(T_1) = 2$. Supposons à présent que sur l'instance I_{k-1}, la solution T_{k-1} renvoyée par l'algorithme (privé de sa dernière itération) soit une chaîne hamiltonienne de l_{k-1} à m_{k-1} de valeur $val(T_{k-1}) = (k+2)2^{k-2} - 2$; nous démontrons que ce résultat tient pour I_k, c'est-à-dire, que $val(T_k) = (k+3)2^{k-1} - 2$. Si l'algorithme commence par le sommet $l_k = l_{k-1}^1$, alors par construction de I_k, l'algorithme va produire un tour partiel sur I_{k-1} qui, par hypothèse de récurrence, consiste en une chaîne hamiltonienne T_{k-1}^1 sur les sommets de G_{k-1}^1, de valeur $(k+2)2^{k-2} - 2$, partant de l_{k-1}^1 et arrivant en m_{k-1}^1. Ensuite, tous les sommets non encore visités (c'est-à-dire, ceux de la copie G_{k-1}^2 et le sommet m_k) sont à une distance d'au moins $2^{k-2}+1$ de m_{k-1}^1 ; or, l_{k-1}^2 atteint cette borne (considérer la distance de m_{k-1}^1 à r_{k-1}^1, plus l'arête $[r_{k-1}^1, l_{k-1}^2]$). Ainsi, l'algorithme va d'abord visiter l_{k-1}^2, puis (par hypothèse de récurrence) parcourir la copie G_{k-1}^2 selon la chaîne hamiltonienne T_{k-1}^2, jusqu'au sommet m_{k-1}^2 ; il achève finalement son parcours en empruntant l'arête $[m_{k-1}^2, m_k]$, dont la distance est de $2^{k-2} + 1$. Pour reconstituer le parcours complet, $T_k = T_{k-1}^1 \cup \{[m_{k-1}^1, l_{k-1}^2]\} \cup T_{k-1}^2 \cup \{[m_{k-1}^2, m_k]\}$; il s'agit d'une chaîne hamiltonienne de l_k à m_k, de valeur :

$$val(T_k) = 2\left((k+2)2^{k-2} - 2\right) + 2\left(2^{k-2} + 1\right) = (k+3)2^{k-1} - 2$$

Ce qui conclut la preuve par récurrence. ∎

THÉORÈME 1.11.– *[HUR 04] Il existe une famille d'instances métriques $I'_k = (K_n, d)$ pour laquelle l'algorithme plus proche voisin produit un tour T'_k dont la valeur vérifie $val(T'_k) = \frac{1}{4}(\lceil \log_2 n \rceil + 3) opt_{min\Delta TSP}(I'_k)$.*

Preuve. De nouveau, l'instance I'_k est construite à partir d'un graphe connexe G' que l'on complète en affectant les distances à la valeur d'un plus court chemin (en nombre de sommets) sur G' ; il est aisé de vérifier qu'ainsi construite, l'instance I'_k est métrique. Le graphe G' est construit à partir de G_k, auquel est ajouté un nouveau sommet s que l'on relie à l_k et r_k. Si l'algorithme commence son parcours par le sommet l_k, alors il va d'abord emprunter tous les sommets de G_k, puisque le sommet s sera à chaque itération toujours au moins aussi éloigné du dernier sommet choisi que tout autre sommet non encore visité. Ainsi, s'appuyant sur le lemme 1.6, le tour $T'k$ qui sera construit par l'algorithme sur I'_k est décrit par $T'_k = T_k \cup \{[m_k, s], [s, l_k]\}$ et sa valeur est donnée par :

$$d(T'_k) = (k+3)2^{k-1} - 2 + (2^{k-1} + 1) + 1 = (k+4)2^{k-1} \qquad [1.25]$$

Par construction, G' est hamiltonien sur $n = 2^{k+1}$ sommets ; I'_k admet donc comme solution optimale un tour T^* de valeur $opt_{min\Delta TSP}(I'_k) = 2^{k+1}$. La figure 1.17 décrit les tours T'_k et T^*_k lorsque $k = 3$ (on note en traits pointillés les arêtes qui n'appartiennent pas à G').

D'après l'inégalité [1.25], le rapport classique d'approximation réalisé par la solution T' sur G' est donc de $k + 4 = \log_2 n + 3 = O(\log_2 n)$, ce qui correspond bien au résultat annoncé. ∎

La variation de l'algorithme plus proche voisin consistant à appliquer n fois l'algorithme en partant à chaque fois d'une ville différente et à retourner le meilleure des n tours a été proposée dans [GAV 65]. Nous laissons au lecteur le soin de prouver que cette variante garantit encore au pire des cas des performances proportionnelles à $O(\log_2 n)$.

1.6.2. *Algorithme de la plus proche insertion*

L'algorithme que nous explorons maintenant est également un algorithme constructif, proposé avec diverses variantes dans [KAR 64]. Son principe est d'insérer à chaque itération un nouveau sommet dans un cycle hamiltonien construit sur les sommets déjà insérés.

Algorithme plus proche insertion :

1) Prendre arbitrairement $v \in V(K_n)$ et poser $V \longleftarrow V \setminus \{v\}$;
2) Trouver $v' \in V'$ tel que $d(v, v') = \min\{d(v, z) : z \in V'\}$;
3) Poser $E(T) \longleftarrow \{[v, v'], [v', v]\}$ et $V' \longleftarrow V' \setminus \{v'\}$;
4) Tant que $V' \neq \emptyset$, faire :

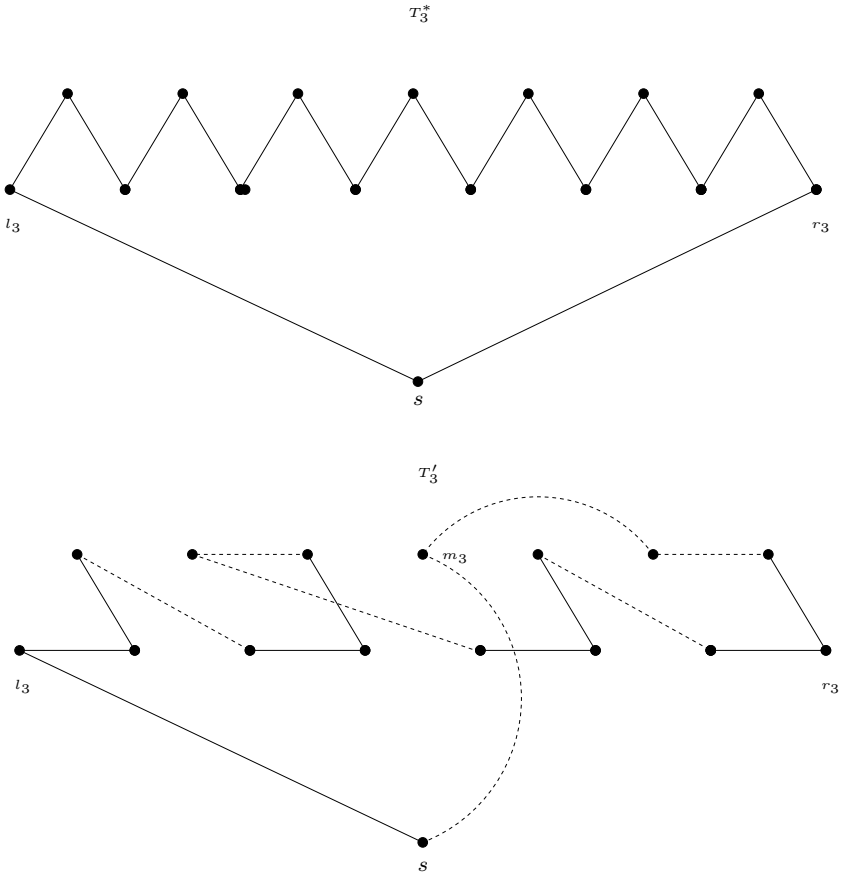

Figure 1.17. *Les solutions T_3' et T_3^* de I_3'*

5) trouver $(v, v') \in V \setminus V' \times V'$ tel que $d(v, v') = \min\{d(x, y) : x \in V \setminus V', y \in V'\}$;

6) trouver $[v, w] \in T'$ telle que $d(v, v') + d(v', w) - d(v, w)$ est minimale ;

7) poser $E(T) \longleftarrow (E(T) \setminus \{[v, w]\}) \cup \{[v, v'], [v', w]\}$ et $V' \longleftarrow V' \setminus \{v'\}$;

8) Renvoyer T.

La figure 1.18 illustre une itération de l'algorithme plus proche insertion.

THÉORÈME 1.12.– *[ROS 77] Pour toute instance métrique $I = (K_n, d)$, l'inégalité suivante est valable :* $val(T) \leq 2(1 - \frac{1}{n})opt_{min\Delta TSP}(I)$.

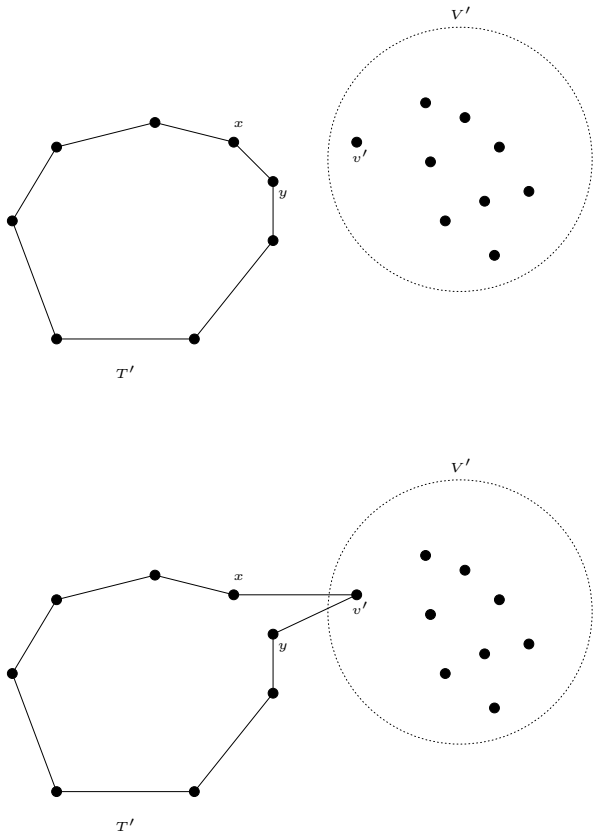

Figure 1.18. *Une itération de l'algorithme*

Preuve. Soit $I = (K_n, d)$ une instance de MIN TSP ; afin de simplifier la preuve, nous numérotons les sommets conformément à l'ordre selon lequel l'algorithme les aura incorporés à la solution. De plus, nous notons par $(E_i)_i$ la suite des sous-ensembles d'arêtes définie par la relation de récurrence suivante :

– $E_2 = \{[v_1, v_2]\}$;

– pour $i \geq 3$, $E_i = E_{i-1} \cup \{[v_i, v_{j_i}]\}$, où $v_{j_i} = \arg\min\{d(v_i, v_j) : j < i\}$.

On vérifie aisément que E_i est un arbre couvrant de poids minimal sur la sous-instance engendrée par $\{v_1, \ldots, v_i\}$. Nous allons montrer par récurrence que si T_i désigne le tour partiel construit par l'algorithme à l'itération i, c'est-à-dire le tour

constitué sur les sommets $\{v_1, \ldots, v_i\}$ (pour $i = 2$, on doit considérer le multigraphe), alors nous avons :

$$\forall i \geq 2,\ val(T_i) \leq 2d(E_i) \qquad [1.26]$$

Pour $i = 2$, puisque $T_2 = \{[v_1, v_2], [v_2, v_1]\}$ et $E_2 = \{[v_1, v_2]\}$, on a immédiatement $val(T_2) = 2d(E_2)$. Supposons à présent que $val(T_{i-1}) \leq 2d(E_{i-1})$, nous allons démontrer que l'inégalité reste valable à l'ordre i. Si $e_1 = [v_{j_i}, x]$ désigne l'arête ajoutée avec $[v_i, v_{j_i}]$ à l'itération i et $e_2 = [x, v_i]$ désigne l'arête qui est retirée de T_{i-1}, l'inégalité triangulaire induit la relation $d(e_1) \leq d(e_2) + d(v_i, v_{j_i})$. De plus, d'après l'itération 6 de l'algorithme, la valeur du tour à l'itération i vérifie $val(T_i) \leq val(T_{i-1}) + (d(v_i, v_{j_i}) + d(e_1) - d(e_2))$. En se servant de ces deux inégalités, nous déduisons la relation $val(T_i) \leq val(T_{i-1}) + 2d(v_i, v_{j_i})$ qui, ajoutée à l'hypothèse de récurrence, permet d'établir l'inégalité [1.26].

Finalement, puisque $T = T_n$, il vérifie $d(T) \leqslant d(E_n)$; or, en retirant au tour optimal T^* une arête de plus grand coût, on obtient un arbre couvrant E' de valeur au plus $d(T^*) - \frac{d(T^*)}{n}$; ainsi, il n'y a plus qu'à remarquer que $d(E') \geqslant d(E_n)$ pour obtenir le résultat souhaité. ∎

Nous laissons le soin au lecteur de prouver que cette inégalité est la meilleure qui soit énoncée sous cette forme.

1.7. Conclusion

Un horizon des algorithmes conçus pour résoudre le problème du voyageur de commerce nous a permis de parcourir une multitude d'approches. Cependant, d'autres voies restent à envisager, dont certaines sont déjà à l'étude. Qu'il s'agisse des méthodes ou des versions mêmes du problème, les pistes ouvertes par les chercheurs ne cessent de se renouveler. Concernant les versions, nous pensons notamment au TSP multicritère, [ANG 03, ANG 05], mais aussi aux instances *on line*, [AUS 95], ou encore stochastiques, [PER 98]. La version multicritère ne travaille plus sur une mais k fonctions de distances $d_1, \ldots\ d_k$: il n'est dès lors plus question de minimiser la longueur d'un tour sur un graphe, mais de trouver des tours qui soient *Pareto-optimaux* ou *non dominés* : un tour T est Pareto-optimal si l'on ne peut trouver de tour T' qui soit strictement meilleur sur un critère (exemple, $d_1(T') < d_1(T)$) sans dégrader strictement un autre critère (exemple, $d_2(T') > d_2(T)$). Dans le cadre on line, les caractéristiques de l'instance sont connues « au fil de l'eau » ; on est donc amené à concevoir progressivement une solution, sans connaissance du graphe final. L'inconnu est également une caractéristique des instances stochastiques puisqu'au lieu de disposer d'une fonction de distance, c'est d'une suite de variables aléatoires que celles-ci sont dotées.

Plus simplement enfin, nous pensons au cas asymétrique où la distance pour aller de v_i à v_j n'est pas nécessairement la même que la distance pour se rendre de v_j en v_i ; dans ce cas plus général, un résultat récent propose une approximation à 8/13 pour MAX TSP, [BLÄ 04]. Concernant les approches, si les voies d'exploration sont de plus en plus nombreuses, il peut néanmoins paraître surprenant que certains des meilleurs algorithmes connus (ou des meilleures évaluations qui en auront été faites) datent d'il y a quelques années : citons d'abord l'approximation à 7/6, [PAP 93] faite en 1993 de la version minimisation du voyageur de commerce où la fonction distance est à valeur dans $\{1, 2\}$ (cas particulier du cas métrique) ; considérons encore l'algorithme de Christofides que nous avons présenté dans ce chapitre et qui date tout de même de 1976.

Hormis l'algorithme de Little, nous avons peu parlé de la recherche exhaustive et des métaheuristiques. Pourtant, de nombreuses autres méthodes par séparation et évaluation ont été développées, ainsi que des tentatives de résolution par la recherche tabou, par la mise en place d'algorithmes génétiques ou encore par des réseaux neuronaux. Il faut dire que pour ces approches, les techniques sont à la fois fort pointues et complexes car souvent dédiées à une famille particulière de problèmes issue du monde industriel, ce qui rend leur évaluation empirique d'autant plus délicate. La bibliographie électronique TSPBIB à l'adresse http ://www.densis.fee.unicamp.br/∼moscato/TSPBIB_home.html constitue, notamment pour ces méthodes, une excellente source d'information.

En conclusion, on se laisse à dire que tout a été fait pour le TSP, des méthodes les plus simples aux plus sophistiquées (comme en témoigne l'excellent parcours de son histoire qui est proposé dans [LAW 85]) et que néanmoins, aucune de ces approches n'a eu, dès lors, raison du coriace TSP. Malgré toute l'application et l'ingéniosité dont font preuve les chercheurs, le voyageur de commerce, le vrai, n'a en conséquence que son GPS pour se guider.

1.8. Bibliographie

[AAR 97] AARTS E., LENSTRA J., Eds., *Local search in combinatorial optimization*, John Wiley & Sons, Chichester, 1997.

[ANG 03] ANGEL E., BAMPIS E., GOURVÈS L., « Approximating the Pareto curve with local search for the bicriteria TSP(1,2) problem », *Theoretical Computer Science*, vol. 310, p. 135-146, 2003.

[ANG 05] ANGEL E., BAMPIS E. , GOURVÈS L., MONNOT J., « (Non)-Approximability for the multi-criteria TSP(1,2) », *Fundamentals of Computation Theory, 15th International Symposium, FCT'05, LNCS*, vol. 3623, p. 329-340, 2005.

[ARO 98] ARORA S., « Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems », *J. ACM*, vol. 45, p. 753-782, 1998.

[AUS 95] AUSIELLO G., FEUERSTEIN E., LEONARDI S., STOUGIE L., TALAMO M., « Competitive Algorithms for the Traveling Salesman », *Algorithms and Data Structures, 4th International Workshop, WADS '95, LNCS*, vol. 955, p. 206-217, 1995.

[BAR 02] BARVINOK A., GIMADI E. K., SERDYUKOV A. I., « The maximum traveling salesman problem », GUTIN G., PUNNEN A. P. (EDS.) *The Traveling Salesman Problem and Its Variations*, Kluwer, Dordtecht, 2002.

[BEN 99] BENDER M. A., CHEKURI C., « Performance guarantees for the TSP with a parametrized triangle inequality », *Algorithms and Data Structures, 6th International Workshop, WADS '99, LNCS*, vol. 1663, p. 80-85, 1999.

[BEN 92] BENTLEY J. L., « Fast algorithms for geometric traveling salesman problems », *ORSA Journal on Computing*, vol. 4, p. 387-411, 1992.

[BER 73] BERGE C., *Graphs and hypergraphs*, North Holland, Amsterdam, 1973.

[BLÄ 04] BLÄSER M., « An $\frac{8}{13}$-approximation algorithm for the asymmetric maximum TSP », *J. of algorithm*, vol. 50, p. 23-48, 2004.

[BÖC 00] BÖCKENHAUER H.-J., HROMKOVIČ J., KLASING R., SEIBERT S., UNGER W., « Approximation algorithms for the TSP with sharpened triangle inequality », *Information Processing Letters*, vol. 75, p. 133-138, 2000.

[CHE 05] CHEN Z. Z., OKAMOTO Y., WANG L., « Improved deterministic approximation algorithms for Max TSP », *Information Processing Letters*, vol. 95, p. 333-342, 2005.

[CHR 76] CHRISTOFIDES N., Worst-case analysis of a new heuristic for the traveling salesman problem, Technical report 338, Grad. School of Industrial Administration, Carnegie-Mellon University, Pittburgh, 1976.

[CRO 58] CROES G. A., « A method for solving traveling-salesman problems », *Operations Research*, vol. 5, p. 791-812, 1958.

[FIS 95] FISCHER S. T., « A note on the complexity of local search problems », *Information Processing Letters*, vol. 53, p. 69-75, 1995.

[FIS 79] FISHER M. L., NEMHAUSER G. L., WOLSEY L. A., « An analysis of approximations for finding a maximum weight Hamiltonian circuit », *Oper. Res.*, vol. 27, p. 799-809, 1979.

[GAR 79] GAREY M. R., JOHNSON D. S., *Computers and intractability. A guide to the theory of NP-completeness*, W. H. Freeman, San Francisco, 1979.

[GAR 85] GARFINKEL R. S., « Motivation and modeling », E. L LAWLER, J. K. LENSTRA, A. H. G. RINNOOY KAN AND D. B. SHMOYS (EDS.) *The Traveling Salesman Problem : a guided tour of Combinatorial Optimization*, Wiley ,Chichester, p. 17-36, 1985.

[GAV 65] GAVETT J., « Three heuristics rules for sequencing jobs to a single production facility », *Management Sci.*, vol. 11, p. 166-176, 1965.

[GON 95] GONDRAN M., MINOUX M., *Graphes et Algorithmes*, Eyrolles, Paris, 1995.

[GUT 99] GUTIN G., « Exponential neighbourhood local search for the traveling salesman problem », *Comput. Oper. Res.*, vol. 26, p. 313-320, 1999.

[HAR 84] HARTVIGSEN D., Extensions of Matching Theory, PhD thesis, Carnegie-Mellon University, 1984.

[HAR 99] HARTVIGSEN D., « The square-free 2-factor problem in bipartite graphs », *Integer Programming and Combinatorial Optimization, 7th International IPCO'99 Conference, LNCS*, vol. 1610, p. 234-241, 1999.

[HAS 00] HASSIN R., RUBINSTEIN S., « Better approximations for Max TSP », *Information Processing Letters*, vol. 75, p. 181-186, 2000.

[HAS 01] HASSIN R., KHULLER S., « z-Approximations », *J. of Algorithms*, vol. 41, p. 429-442, 2001.

[HAS 02] HASSIN R., RUBINSTEIN S., « A 7/8-approximation algorithm for metric Max TSP », *Information Processing Letters*, vol. 81, p. 247-251, 2002.

[HEL 62] HELD M., KARP R. M., « A dynamic programming approach to sequencing problems », *SIAM J. Appl. Math.*, vol. 10, p. 196-210, 1962.

[HUR 04] HURKENS C. A. J., WOEGINGER G. J., « On the nearest neighbor rule for the traveling salesman problem », *Oper. Res. Let.*, vol. 32, p. 1-4, 2004.

[JOH 85] JOHNSON D. S., PAPADIMITRIOU C. H., « Performance guarantees for heuristics », E. L LAWLER, J. K. LENSTRA, A. H. G. RINNOOY KAN AND D. B. SHMOYS (EDS.) *The Traveling Salesman Problem : a guided tour of Combinatorial Optimization*, Wiley, Chichester, p. 145-180, 1985.

[JOH 97] JOHNSON D. S., MCGEOCH L. A., « The traveling salesman problem : a case study », AARTS E., LENSTRA J. K. (EDS.) *Local search in combinatorial optimization*, Wiley, Chichester, 1997.

[JOH 02] JOHNSON D. S., MCGEOCH L. A., « Experimental Analysis of Heuristics for the STSP », GUTIN G., PUNNEN A. P. (EDS.), *The Traveling Salesman Problem and Its Variations*, Kluwer, Dordtecht, 2002.

[KAR 64] KARG L. L., THOMPSON G. L., « A heuristic approach to traveling salesman problems », *Management Sci.*, vol. 10, p. 225-248, 1964.

[KIR 99] KIRÁLY Z., C_4-free 2-factors in bipartite graphs, Technical report EGRES Budapest, Hungary, 1999.

[LAW 85] LAWLER L., LENSTRA J. K., RINNOOY KAN A. H. G., SHMOYS D. B., *The Traveling Salesman Problem : a guided tour of Combinatorial Optimization*, Wiley, Chichester, 1985.

[LIN 73] LIN S., KERNIGHAN B. W., « An effective heuristic algorithm for the traveling salesman problem », *Oper. Res.*, vol. 21, p. 498-516, 1973.

[LIT 63] LITTLE J. D. C., MURTY K. G., SWEENEY D. W., KAREL C., « An algorithm for the traveling salesman problem », *Oper. Res.*, vol. 11, p. 972-989, 1963.

[MON 02a] MONNOT J., « Approximation results toward Nearest Neighbor heuristic », *Yugoslav Journal of Operations Research*, vol. 12, n°1, p. 11-16, 2002.

[MON 02b] MONNOT J., « Differential approximation results for the Traveling Salesman and related Problems », *Information Processing Letters*, vol. 82(5), p. 229-235, 2002.

[MON 02c] MONNOT J., PASCHOS V. T., TOULOUSE S., « Approximation Algorithms for the traveling salesman problem », *Math. Models of Op. Res.*, vol. 56, p. 387-405, 2002.

[MON 03] MONNOT J., PASCHOS V. T., TOULOUSE S., *Approximation polynomiale des problèmes NP-difficiles : optima locaux et rapport différentiel*, Hermès, Paris, 2003.

[NIC 67] NICHOLSON T. A. J., « A sequential method for discrete optimization problems and its application to the assignment, Traveling Salesman, and three machine scheduling problems », *J. Inst. of Maths. Applics*, vol. 3, p. 362-375, 1967.

[PAP 77] PAPADIMITRIOU C. H., STEIGLITZ K., « On the complexity of local search for the traveling salesman problem », *SIAM J. Comput.*, vol. 6, p. 76-83, 1977.

[PAP 78] PAPADIMITRIOU C. H., STEIGLITZ K., « Some examples of difficult traveling salesman problems », *Oper. Res.*, vol. 26, p. 434-443, 1978.

[PAP 93] PAPADIMITRIOU C. H., YANNAKAKIS M., « The traveling salesman problem with distances one and two », *Math. Oper. Res.*, vol. 18, p. 1-11, 1993.

[PAP 98] PAPADIMITRIOU C. H., STEIGLITZ K., *Combinatorial optimization : algorithms and complexity*, Dover, New York, 1998.

[PER 98] PERCUS A. G., MARTIN O. C., « Scaling universalities of kth-nearest neighbor distances on closed manifolds », *Advances in Applied Mathematics*, vol. 21, p. 424-436, 1998.

[ROS 77] ROSENKRANTZ D. J., STEARNS R. E., LEWIS II P. M., « An analysis of several heuristics for the traveling salesman problem », *SIAM J. Comp.*, vol. 6, p. 563-581, 1977.

[SER 84] SERDYUKOV A. I., « An algorithm with an estimate for the traveling salesman problem of the maximum », *Upravlyaemye Sistemy (en Russe)*, vol. 25, p. 80-86, 1984.

[VOR 80] VORNBEGER O., Easy and hard cycle cover, Technical report, univerität Paderborn, Allemagne, 1980.