L'opérateur PUSH pour la résolution approchée du problème de la clique de poids maximum

Yi Zhou[1], Jin-Kao Hao[1,2], Adrien Goëffon[1]

[1] LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers Cedex 01, France
{zhou,hao,goeffon}@info.univ-angers.fr
[2] Institut Universitaire de France, Paris, France

Mots-clés : *MWCP, clique maximum, opérateur PUSH, recherche locale, recherche tabou.*

1 Présentation du problème

Soit $G = (V, E, w)$ un graphe non orienté, où V et $E \subseteq V^2$ dénotent respectivement l'ensemble de sommets et d'arêtes, et $w : V \rightarrow \mathbb{R}_+^*$ est une fonction de poids affectant une valeur positive à chaque sommet $v \in V$. Une clique $C \subseteq V$ de G est un sous-ensemble de sommets tel que le sous-graphe induit est complet, c'est-à-dire $\{u, v\} \in C \rightarrow (u, v) \in E$. $W(C) := \sum_{v \in C} w(v)$ est le poids de C. Le problème de la clique de poids maximum MWCP (*Maximum Weight Clique Problem*) consiste à déterminer une clique d'un graphe G de poids (ou valeur) maximum, soit $\arg\max_{C \subseteq V} W(C)$. MWCP généralise le problème classique de la clique maximum (MCP), et trouve des applications dans des domaines variés [5]. Puisque le problème de décision associé à MCP est NP-complet [2], le problème généralisé MWCP est au moins ausi difficile que MCP, ce qui rend sa résolution ardue dans le cas général.

2 L'opérateur de recherche locale PUSH

Nous avons récemment introduit un opérateur de transformation de clique appelé *PUSH* [3], dédié à MWCP mais généralisant les opérateurs classiques pour MCP et MWCP. *PUSH* ajoute à une clique un sommet sélectionné parmi un ensemble de candidats CPS (*candidate push set*), et retire les éventuels sommets de la précédente clique afin de maintenir la faisabilité de la nouvelle. Non seulement l'opérateur *PUSH* généralise les opérateurs traditionnels d'ajout et d'échange de sommets (*ADD*, *SWAP*), mais offre également la possibilité de définir des opérateurs de transformation additionnels. En effet, des opérateurs de recherche locale personnalisés peuvent être obtenus en contraignant l'ensemble CPS considéré par *PUSH*, afin d'intensifier ou diversifier la recherche. *PUSH* permet également de générer des mouvements améliorants qui ne peuvent pas toujours être permis au moyen des opérateurs de base *ADD* et *SWAP*, ce qui offre des opportunités supplémentaires à un algorithme de recherche locale d'atteindre des solutions de très bonne qualité.

3 Résultats expérimentaux

Pour évaluer l'intérêt d'utiliser l'opérateur *PUSH*, nous avons implémenté deux algorithmes de type *restart tabu search* (ReTS-I and ReTS-II), chacun basé sur une gestion différente des mouvements PUSH applicables à chaque itération. Ainsi, ReTS-I ne contraint pas l'ensemble CPS et envisage toujours le plus grand ensemble de voisins possibles. ReTS-II est basé sur trois ensembles CPS de différentes priorités (mouvements PUSH strictements améliorants, échanges de sommets équivalents et détériorants, autres mouvements PUSH). Les deux algorithmes partagent la même stratégie de perturbation, à savoir un mécanisme de *restart* probabiliste qui relance la recherche depuis une nouvelle solution, dérivée ou non de la solution courante). Les

TAB. 1 – Comparaisons de ReTS-I et ReTS-II avec 3 algorithmes référence sur 27 instances DIMACS et BHOSLIB (100 éxécutions par algorithme et par instance). La colonne Δ indique l'écart entre la meilleure solution trouvée et la meilleure solution connue au préalable en termes de valeur (BKV), ainsi que la fréquence d'apparition du score maximal sur les 100 exécutions. Les valeurs Δ positives pour l'instance frb53-24-3 indiquent une amélioration de la meilleure borne connue.

		ReTS-I			ReTS-II			MN/TS[6]			BLS[1]			BQP-PTS[4]		
instances	BKV	Δ	fréq.	tps (s)	Δ	fréq.	tps (s)	Δ	fréq.	tps (s)	Δ	fréq.	tps (s)	Δ	fréq.	tps (s)
C2000.9	10999	0	92	417.56	0	82	474.23	0	72	168.11	0	74	1152.78	0	72	2711.97
MANN_a27	12283	**0**	78	82.77	**0**	99	60.03	-2	1	88.28	-2	16	396.58	-6	4	12264
MANN_a45	34265	**-6**	1	157.98	**-13**	58	357.19	-73	1	390.58	-36	1	929.41	-71	2	17524.05
MANN_a81	111386	**-16**	1	990.02	**-109**	1	477.75	-258	1	832.24	-149	1	2942.54	-249	1	6167.28
p_hat1000-3	8111	0	100	0.19	0	100	0.21	0	96	188.38	0	100	1.78	0	100	0.65
brock800_4	2971	0	31	835.03	0	93	506.41	0	100	49.70	0	100	339.07	0	8	105.35
keller6	8062	0	100	532.74	0	96	929.74	0	5	606.15	0	44	1980.16	0	2	3418.36
frb50-23-1	5494	0	4	154.05	0	4	590.72	0	6	186.62	0	11	1221.72	0	20	1911.49
frb50-23-2	5462	0	9	393.53	0	44	458.97	0	3	14966	0	5	2837.74	0	15	2338.40
frb50-23-3	5486	0	57	358.92	0	87	292.35	0	53	158.71	0	98	537.96	0	100	418.35
frb50-23-4	5454	-1	91	243.84	0	6	548.32	0	9	176.41	0	14	1190.43	0	28	1957.22
frb50-23-5	5498	0	100	118.21	0	94	277.51	0	89	110.85	0	100	388.18	0	100	751.84
frb53-24-1	5670	0	33	349.95	0	58	325.66	0	5	233.22	0	13	1056.82	0	43	981.33
frb53-24-2	5707	0	1	880.86	0	5	415.40	0	6	145.22	0	3	147.65	0	25	1265.70
frb53-24-3	5640	**15**	3	417.69	**15**	3	457.64	0	15	215.79	0	48	984.53	0	90	1486.24
frb53-24-4	5714	0	4	421.52	0	7	402.50	0	7	449.39	0	13	1604.50	0	25	1753.36
frb53-24-5	5659	0	1	777.93	0	5	381.24	0	5	294.00	0	4	278.91	0	6	2802.83
frb56-25-1	5916	0	59	344.18	0	51	428.24	0	3	308.90	0	5	1764.87	0	19	1035.00
frb56-25-2	5886	0	9	516.06	0	37	470.10	-14	1	73.25	0	1	1013.85	0	3	1428.18
frb56-25-3	5859	0	1	450.99	-5	1	30.19	0	1	191.93	0	1	101.48	0	5	1756.22
frb56-25-4	5892	0	2	477.79	-7	2	449.13	0	3	104.58	0	12	1256.9	0	5	1756.22
frb56-25-5	5853	-12	1	354.28	0	2	514.91	0	1	322.70	0	1	4386.6	0	1	3549.57
frb59-26-1	6591	0	20	521.08	0	32	432.31	0	3	166.20	0	17	1435.99	0	67	2228.21
frb59-26-2	6645	0	13	505.28	0	25	660.05	0	3	212.49	0	13	1834.93	0	40	1820.56
frb59-26-3	6608	0	1	973.94	0	25	455.84	0	1	232.77	0	1	507.93	0	1	2561.16
frb59-26-4	6592	0	71	377.92	0	18	541.34	0	1	318.39	0	6	952.34	0	5	3322.64
frb59-26-5	6584	0	3	320.54	0	9	399.46	0	1	161.47	0	5	1512.09	0	9	747.80

algorithmes ont été testés sur 142 instances de problèmes issus des ensembles de benchmarks DIMACS, BHOSLIB, and *Winner Determination Problem*. Sur la Table 1, nous constatons que ReTS-I et ReTS-II atteignent sur 4 instances de meilleurs résultats que les 3 algorithmes référence utilisés pour comparaison. Bien que les instances MANN_aXX soient reconnues comme particulièrement difficiles à résoudre par des heuristiques, ReTS-I et ReTS-II ont pu déterminer une solution optimale sur MANN_a27 et de meilleures solutions approchées sur MANN_a45 et MANN_a81. Notons que les algorithmes BLS et BQP-PTS utilisent des temps de calcul plus importants. Ainsi, dans des tests additionnels, nous avons étendu le nombre d'itérations par exécution à 1.6×10^8 pour ReTS-I et ReTS-II (comme défini pour BLS), ce qui leur a suffit pour atteindre les meilleures bornes connues (BKV) sur toutes les instances testées.

Nous avons présenté dans ce résumé un algorithme tabou basé sur un opérateur de transformation de clique original nommé *PUSH*. D'après nos expérimentations, les deux versions proposées de l'algorithme obtiennent des résultats équivalents ou meilleurs que les principales heuristiques développées pour résoudre MWCP. La généralité de l'opérateur *PUSH* devrait lui permettre d'être intégrée aux autres heuristiques afin d'améliorer leurs performances.

Références

[1] U. Benlic, J.K. Hao, Breakout local search for maximum clique problems, Computers & Operations Research 40 (1) (2013) 192–206.

[2] R. M. Karp, Reducibility among combinatorial problems, Springer, 1972.

[3] Y. Zhou, J.K. Hao, A. Goëffon. PUSH : a generalized operator for the maximum weight clique problem. European Journal of Operational Research 257(1) (2017) 41–54.

[4] Y. Wang, J.K. Hao, F. Glover, Z, Lü, Q. Wu, Solving the maximum vertex weight clique problem via binary quadratic programming, Journal of Combinatorial Optimization 32(2) (2016) 531–549.

[5] Q. Wu, J.K. Hao, A review on algorithms for maximum clique problems, European Journal of Operational Research 242 (3) (2015) 693–709.

[6] Q. Wu, J.K. Hao, F. Glover, Multi-neighborhood tabu search for the maximum weight clique problem, Annals of Operations Research 196 (1) (2012) 611–634.