Supplementary Material for Penalty Weights in QUBO formulations of Permutation Problems

Mayowa Ayodele[1][0000−0003−0854−4777]

Fujitsu Research of Europe, London, UK
mayowa.ayodele@fujitsu.com

Abstract. This is a supplementary material for paper titled Penalty Weights in QUBO formulations of Permutation Problems. It contains full results of experiments which could not fit in the paper. Experiments are those relating to runs of the third generation Digital Annealer [**?**]

1 Penalty Weights

The aim of this study is to derive methods of setting α in Eq. (**??**) such that the optimal solution to the penalised objective function is the optimal solution of the original constrained problem. We do this without problem specific knowledge but use information captured in the QUBO matrices representing the cost and constraint functions. This is shown in Eq. (1), $c(y)$ is used to denote the cost function of the optimal solution y. S is the solution space of infeasible solutions. Note that $g(x)$ produces a non-negative value, $g(x) = 0$ if the solutions are feasible but $g(x) > 0$ for infeasible solutions. The value of $g(x)$ increases according the degree of constraint violation.

$$c(y) < c(x) + \alpha \times g(x) \;\; \forall \; x \in S \tag{1}$$

Eq. (1) implies that a valid penalty weight α is one that satisfies Eq. (2)

$$\alpha > \max_{x \in S} \left(\frac{c(y) - c(x)}{g(x)} \right) \tag{2}$$

In the rest of this study, C and G are used to denote the QUBO matrices representing $c(x)$ and $g(x)$ respectively. Note that Q, which is the QUBO matrix optimised by the solver, can be derived by aggregating the matrices (i.e. $Q = C + \alpha \times G$), where $\alpha \geq 1$. The methods of generating penalty weights used in this study are described as follow:

UB: A common method of setting penalty weights is based on the Upper Bound (UB) of the Objective function. The UB of C used in this study is presented in Eq. (3). This is a valid upper bound for problems with all positive QUBO coefficients. We note that a solution consisting of all 1s is an infeasible solution but it gives an estimate of how large the objective function could be.

$$\text{UB} = z^T C z, \;\; z_i = 1 \; \forall \; i \; \in [1, n] \tag{3}$$

MQC: The Maximum QUBO Coefficient (MQC) which also corresponds to the maximum distance between any two cities in the TSP has been used as penalty weights in previous study [**?**]. MQC is defined in Eq. (4).

$$\text{MQC} = \max_{i=1}^{n} \max_{j=1}^{n} \left(abs\left(C_{i,j} \right) \right) \tag{4}$$

VLM: This is the method proposed by Verma and Lewis in [**?**]. For a 1-flip solver like the DA, any variable x_i can be flipped from 0 to 1 and vice versa at each iteration of the algorithm. VLM focuses on deriving a good estimate for the numerator of Eq. (2), i.e. $(c(y) - c(x))$. The method estimates the amount of gain/loss in objective function that can be achieved by either turning a bit on or off. They do not consider the denominator (i.e $g(x)$) which the authors recognise will be hard to estimate without complete enumeration. Method of generation α using VLM is shown in Eq. (5)

$$W^c = \left\{ -C_{i,i} - \sum_{\substack{j=1 \\ j \neq i}}^{n} \min\{C_{i,j}, 0\},\ C_{i,i} + \sum_{\substack{j=1 \\ j \neq i}}^{n} \max\{C_{i,j}, 0\}\ \forall\ i \in [1, n] \right\} \tag{5}$$

$$\alpha = \text{VLM} = \max_{i=1}^{n} W_i^c \tag{6}$$

MOMC: We propose an amendment to the VLM method [**?**]. To differentiate this method from the one above (VLM), we refer to this method as the Maximum change in Objective function divided by Minimum Constraint function of infeasible solutions (MOMC). We note that $g(x)$ is not considered in Eq. (5). VLM can be reduced such that α is still valid, if we know the minimum constraint function ($g(x)$) of any infeasible solution. This can be computed from G by estimating the minimum change in energy that is greater than 0 as shown in Eq. (7).

$$W^g = \left\{ -G_{i,i} - \sum_{\substack{j=1 \\ j \neq i}}^{n} \min\{G_{i,j}, 0\},\ G_{i,i} + \sum_{\substack{j=1 \\ j \neq i}}^{n} \max\{G_{i,j}, 0\}\ \forall\ i \in [1, n] \right\} \tag{7}$$

$$\gamma = \min_{\substack{i=1 \\ W_i^g > 0}}^{n} W_i^g \tag{8}$$

For permutation problems represented as two-way one-hot, $g(x)$ of any solution that is a flip away from any feasible solution is 2 (i.e. $\gamma = 2$). Method of generation α using the proposed MOMC is presented in Eq. 9 where W^c and γ are derived as shown in Eqs. (5) and (8)

$$\alpha = \text{MOMC} = \frac{\text{VLM}}{\gamma} = \frac{\max_{i=1}^{n} W_i^c}{2} \tag{9}$$

MOC: We propose another amendment to the VLM method. The method presented here is derived by selecting the Maximum value derived from dividing each change in Objective function with the corresponding change in Constraint function (MOC). Method of generation α using the proposed MOC is presented in Eq. (10) where W^c and W^g are derived as shown in Eqs. (5) and (7)

$$\alpha = \text{MOC} = \max_{\substack{i=1 \\ W_i^g > 0}}^{n} abs\left(\frac{W_i^c}{W_i^g}\right) \tag{10}$$

Problem Instances		Optimal	Average Energy (Fitness)					Standard Deviation Fitness				
			MOC	MOMC	MQC	UB	VLM	MOC	MOMC	MQC	UB	VLM
QAP	had12	1,652	1,652	1,652		1,652	1,652	0.00	0.00		0.00	0.00
	had14	2,724	2,724	2,724		2,724	2,724	0.00	0.00		0.00	0.00
	had16	3,720	3,720	3,720		3,720	3,720	0.00	0.00		0.00	0.00
	had18	5,358	5,358	5,358		5,358	5,358	0.00	0.00		0.00	0.00
	had20	6,922	6,922	6,922		6,922	6,922	0.00	0.00		0.00	0.00
	rou12	235,528	235,528	235,528		235,528	235,528	0.00	0.00		0.00	0.00
	rou15	354,210	354,210	354,210		354,210	354,210	0.00	0.00		0.00	0.00
	rou20	725,522	725,522	725,522		725,522	725,522	0.00	0.00		0.00	0.00
	tai40a	3,139,370	3,141,702	3,141,702		3,141,702	3,141,702	0.00	0.00		0.00	0.00
	tai40b	637,250,948	637,250,948	637,250,948		637,250,948	637,250,948	0.00	0.00		0.00	0.00
TSP	bayg29	1,610	1,610	1,610	1,610	1,610	1,610	0.00	0.00	0.00	0.00	0.00
	bays29	2,020	2,020	2,020	2,020	2,020	2,020	0.00	0.00	0.00	0.00	0.00
	berlin52	7,542	7,708	7,998	7,765	7,682	7,856	0.00	0.00	0.00	0.00	0.00
	brazil58	25,395	25,758	25,795	26,241	25,783	25,783	0.00	0.00	0.00	0.00	0.00
	dantzig42	699	699	699	699	699	699	0.00	0.00	0.00	0.00	0.00
	fri26	937	937	937	937	937	937	0.00	0.00	0.00	0.00	0.00
	gr17	2,085	2,085	2,085	2,085	2,085	2,085	0.00	0.00	0.00	0.00	0.00
	gr21	2,707	2,707	2,707	2,707	2,707	2,707	0.00	0.00	0.00	0.00	0.00
	gr24	1,272	1,272	1,272	1,272	1,272	1,272	0.00	0.00	0.00	0.00	0.00
	st70	675	691	691	685	693	691	0.45	0.00	0.00	0.00	0.73

Table 1. DA3: Average and standard deviation fitness within $0.03m$ seconds time limit

Problem Instances		Average Time to Solution					Standard Deviation Time to Solution				
		UB	MQC	VLM	MOMC	MOC	UB	MQC	VLM	MOMC	MOC
QAP	had12	0.20		0.20	0.20	0.21	0.01		0.01	0.01	0.01
	had14	0.22		0.22	0.22	0.22	0.02		0.02	0.02	0.01
	had16	0.25		0.25	0.25	0.25	0.02		0.01	0.01	0.02
	had18	0.31		0.31	0.31	0.31	0.01		0.01	0.01	0.01
	had20	0.38		0.37	0.37	0.38	0.01		0.01	0.01	0.01
	rou12	0.20		0.20	0.20	0.20	0.02		0.01	0.02	0.02
	rou15	0.23		0.23	0.23	0.22	0.02		0.01	0.02	0.02
	rou20	0.37		0.37	0.37	0.37	0.02		0.02	0.02	0.01
	tai40a	19.08		19.07	19.03	5.60	0.16		0.14	0.17	0.08
	tai40b	2.43		2.44	2.43	4.77	0.04		0.04	0.05	0.06
TSP	bayg29	0.47	1.34	0.47	0.48	0.47	0.01	0.07	0.02	0.02	0.02
	bays29	1.34	3.57	1.34	1.34	1.33	0.07	0.14	0.06	0.06	0.06
	berlin52	70.47	64.81	68.17	17.27	53.25	0.92	0.70	1.03	0.25	0.61
	brazil58	73.48	91.83	73.73	49.95	20.04	0.65	1.09	0.59	0.70	0.25
	dantzig42	14.41	19.86	14.50	14.54	45.13	0.35	0.29	0.27	0.16	0.68
	fri26	0.38	0.38	0.38	0.37	0.38	0.01	0.01	0.02	0.02	0.01
	gr17	0.22	0.22	0.22	0.22	0.21	0.01	0.01	0.01	0.01	0.01
	gr21	0.27	0.26	0.26	0.26	0.26	0.02	0.02	0.02	0.02	0.02
	gr24	0.61	0.61	0.61	0.61	0.61	0.03	0.03	0.03	0.03	0.03
	st70	23.60	136.56	124.71	142.34	136.04	0.29	0.76	43.41	0.66	26.41

Table 2. DA3: Average and standard deviation of run time (in seconds) to find best solution within $0.03m$ seconds time limit