BASAR:Black-box Attack on Skeletal Action Recognition
Supplemental Document

Yunfeng Diao[1,2]*, Tianjia Shao[3]†, Yong-Liang Yang[4], Kun Zhou[3], He Wang[1]
[1]University of Leeds, UK [2]Southwest Jiaotong University, China
[3]State Key Lab of CAD&CG, Zhejiang University, China [4] University of Bath, UK
dyf@my.swjtu.edu.cn, tjshao@zju.edu.cn, y.yang@cs.bath.ac.uk, kunzhou@zju.edu.cn, h.e.wang@leeds.ac.uk

1. Additional Experimental Details

1.1. Implement Details and Experimental Settings

We first give details about the Random Exploration and Aimed Probing. For easy reference, the random exploration is reformulated in Eq. 1:

$$\begin{aligned}&\widetilde{\mathbf{x}} = \mathbf{x}' + \mathbf{W}\Delta,\\ &\text{where } \Delta_* = \mathbf{R}_* - (\mathbf{R}_*^T\mathbf{d}_*)\mathbf{d}_*,\ \mathbf{d}_* = \frac{\mathbf{x}_* - \mathbf{x}'_*}{\|\mathbf{x}_* - \mathbf{x}'_*\|},\\ &\mathbf{R}_* = \lambda\frac{\mathbf{r}}{\|\mathbf{r}\|}\|\mathbf{x}_* - \mathbf{x}'_*\|, r \in N(0, \mathbf{I}),\end{aligned} \tag{1}$$

where $\widetilde{\mathbf{x}}$ is the new perturbed sample, $\mathbf{x}$ and $\mathbf{x}'$ are the attacked motion and current adversarial sample. We use joint positions and the subscript $*$ indicates either the x, y, or z joint coordinate. The update on $\mathbf{x}'$ is Δ weighted by $\mathbf{W}$ - a diagonal matrix with joint weights. Δ_* controls the direction and magnitude of the update, and depends on two variables $\mathbf{R}_*$ and $\mathbf{d}_*$. $\mathbf{d}_*$ is the directional vector from $\mathbf{x}'$ to $\mathbf{x}$. $\mathbf{R}_*$ is a random directional vector sampled from a Normal distribution $N(0, \mathbf{I})$ where $\mathbf{I}$ is an identity matrix, $\mathbf{I} \in R^{z\times z}$, $z = mn/3$, m is the number of Dofs in one frame and n is the total frame number. This directional vector is scaled by $\|\mathbf{x}_* - \mathbf{x}'_*\|$ and λ.

The aimed probing is reformulated by Eq. 2:

$$\widetilde{\mathbf{x}} = \mathbf{x}' + \beta(\mathbf{x} - \mathbf{x}'), \tag{2}$$

where β is a forward step size that can also be dynamically adjusted. β is decreased by half to conduct the aimed probing again if $\widetilde{\mathbf{x}}$ is not adversarial; otherwise, β is doubled, then we enter the next sub-routine.

In random exploration, we aim to find an adversarial sample that is closer to $\mathbf{x}$. However, as the shape of the local space is unknown and highly nonlinear, we do sampling to exploit it. Therefore, we execute multiple random explorations instead of only one to get q intermediate results in a sub-routine call, and compute the attack success rate. If the rate is less than 40%, λ is reduced by 10% as it means that we are very close to the classification boundary ∂C and λ is too big; if it is higher than 60%, λ is increased by 10%; otherwise we do not update λ.

For targeted attack, we randomly select one adversarial sample from the q intermediate samples to do aimed probing. This is mainly to ensure that the direction of the aimed probing is random. Although multiple samples can be selected, it would incur more computational costs with little gain shown by our preliminary experiments. For untargeted attack, the q results are normally in different classes which we call *adversarial classes*. The attack difficulty varies depending on the choice of samples. Usually the closer the adversarial sample is to the original sample, the easier the attack. Therefore, we randomly select one sample in each adversarial class to conduct aimed probing, then only keep the one that has the smallest distance to the original motion $\mathbf{x}$ after the aimed probing. In the end, when the adversarial sample is near to the original motion, we set a threshold value τ to ensure that λ is not higher than τ. This is to ensure that the attack can eventually converge.

In all experiments, we set $q = 5$. The initial β is set to 0.95. The initial λ is set to 0.2 when attacking SGN model and 0.1 on both STGCN and MSG3D. τ is set to 1.5 on SGN and 0.4 on both STGCN and MSG3D. We set the spinal joint weights to 0 in $\mathbf{W}$, and other joint weights to 1. For untargeted attack, we set $\epsilon = 0.1$ on both HDM05 and NTU, 0.05 on Kinetics. For targeted attack, ϵ is set to 0.5 on HDM05 and both 0.2 on NTU and Kinetics. Considering the optimization speed, it is unrealistic to execute the manifold projection in every iteration. We therefore execute it every 100 iterations on HDM05 and every 250 iterations on NTU and Kinetics.

The adversarial samples are computed using PyTorch on a PC with a NVIDIA GTX 2080Ti GPU and a Xeon Silver 4216 CPU. We also show how different metrics vary based on the number of actual queries BASAR makes to the at-

---

*The research was conducted during the visit to the University of Leeds.
†Corresponding author

tacked model. The evaluation versus number of queries are shown from Fig 1 to Fig 3. Being consistent with our analysis in the paper, compared with STGCN and MSG3D, SGN usually converges faster but it is difficult for BASAR to further improve the adversarial sample as it does on STGCN and MSG3D. We speculate that this is due to the semantic information that SGN uses prevents small perturbations from altering the class labels.

Models		HDM05		NTU		Kinetics	
		Queries	Time	Queries	Time	Queries	Time
STGCN	UA	3636	4	7337	12	7167	28
	TA	8862	15	15724	16	15234	41
MSG3D	UA	3722	6	14640	18	7190	29
	TA	9111	16	23227	30	15416	56
SGN	UA	974	4	623	5	228	10
	TA	277	3	260	4	180	8

Table 1. The averaged number of queries and consuming time(min) for generating an adversarial sample on different models and datasets.

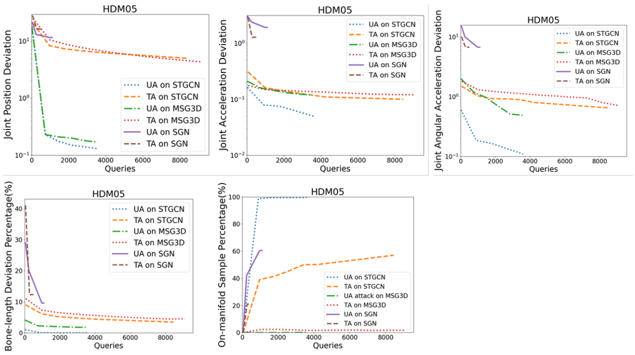


Figure 1. Numerical evaluation versus number of queries on HDM05 with STGCN, MSG3D and SGN. UA/TA refers to Untargeted Attack/Targeted Attack.

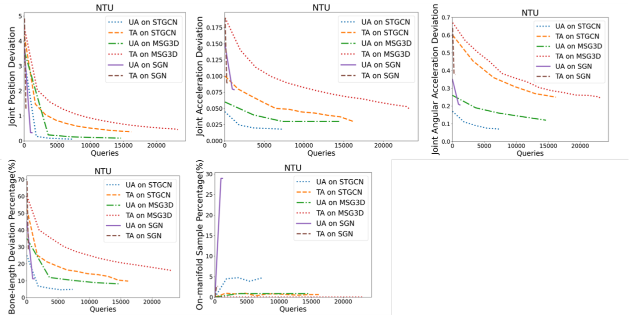


Figure 2. Numerical evaluation versus number of queries on NTU with STGCN, MSG3D and SGN.

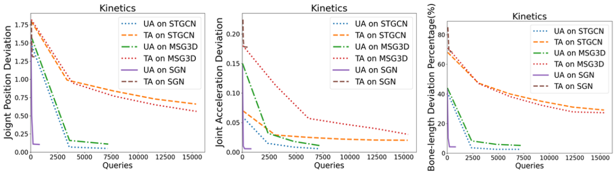


Figure 3. Numerical evaluation versus number of queries on Kinetics with STGCN, MSG3D and SGN.

1.2. Detailed Perceptual Studies

In all 50 subjects (age between 18 and 54), 86% of users are aged under 30 and 88% are male. The users have various background. Around 25% users have research expertise in human activity recognition or adversarial attack; another 20% have general deep learning or computer vision background; 45% people study in engineering (e.g. mechanical, electrical and control). The other users have different arts background. By comparing their performance we found that the age, gender and work/research background actually do not have obvious influence to the results. The results are mainly dependent on the quality of the adversarial samples.

Deceitfulness. This study is to test: whether BASAR visually changes the meaning of the motion and whether the meaning of the original motion is clear to the subjects. In each user study, we randomly choose 45 motions (15 from STGCN, MSG3D and SGN respectively) with the ground truth label and after-attack label for 45 trials. In each trial, the video is played for 6 seconds then the user is asked the question,'which label best describes the motion? and choose Left or Right', with no time limits.

Naturalness. This ablation study is to test whether on-manifold adversarial samples look more natural than off-manifold samples. In this user study, we perform ablation studies to test whether on-manifold adversarial samples look more natural than off-manifold samples. We design two settings: MP and No MP. MP refers to BASAR, with manifold projection. No MP is where the proposed method without manifold projection. In each study, 60 (20 from STGCN, MSG3D and SGN respectively) pairs of motions are randomly selected for 60 trials. Each trial includes one from MP and one from No MP. The two motions are played together for 6 seconds twice, then the user is asked, 'which motion looks more natural? and choose Left, Right or Can't tell', with no time limits.

Indistinguishability. Indistinguishability is the strictest test to see whether adversarial samples by BASAR can survive a side-by-side scrutiny. In each user study, 40 pairs of motion are randomly selected, half from STGCN and half from MSG3D. For each trial, two motions are displayed side by side. The left motion is always the original and the user is told so. The right one can be original (**sensitivity**) or attacked (**perceivability**). The two motions are played together for 6 seconds twice, then the user is asked, 'Do they look same? and choose Yes or No', with no time limits. This user study serves two purposes. **Perceivability** is a direct test on **indistinguishability** while **sensitivity** aims to screen out users who tend to choose randomly. We discard any user data which falls below 80% accuracy on the sensitivity test.

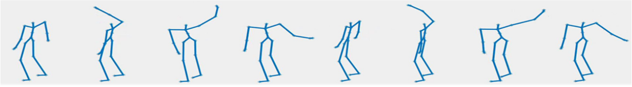
2. Visual Results and Confusion Matrices

The visual results on various datasets and models are shown from Fig. 4 to Fig. 9. As we can see, the adversarial samples on STGCN and MSG3D in general are very hard to be distinguished from the attacked motion. The results on SGN have the same semantic meanings and are almost equally hard to be distinguished from the original motion in untargeted attack. However, when it is targeted attack and the target label is very different from the original label, BASAR sometimes generate adversarial samples with visible differences. We show some failures here (Fig. 6 Bottom and Fig. 9 Bottom.). These adversarial samples might survive a visual examination if shown alone but might not be able to survive a side-by-side comparison with the original motions in our rigorous perceptual studies. This is also consistent with our numerical evaluation.

In NTU, there are actions containing a single person or two persons. We, therefore, attack them separately. In targeted attack, if the attacked motion is a single-person action, the target class is also a single-person action where we randomly select a motion to initiate the attack. Similarly, if the attacked motion is a two-person action, we select a two-person motion. In untargeted attack, we do not need to initiate the attack separately and can rely on BASAR to find the adversarial sample that is closest to the original motion. The confusion matrices across various datasets and models are shown from Fig. 10 to Fig. 15.

In untargeted attack, we find that random attacks easily converge to a few action classes in a dataset. We call them *high-connectivity classes*. For example, actions on STGCN tend to be attacked into 'Jump Jack'(number 20) and 'Kick left front'(21) on HDM05, and into 'Use a fan'(48) on NTU, regardless how they are initialized; Similarly, actions on MSG3D tend to be attacked into 'Cartwheel'(0) and 'Kick right front'(23) on HDM05, and into 'Use a fan'(48) on NTU; actions on SGN tend to be attacked into 'Cartwheel'(0) and 'Jump Jack'(20) on HDM05, and into 'Hopping'(25) on NTU. The theoretical reason is hard to identify but we have the following speculations. Since untargeted attack starts from random motions, it is more likely to find the adversarial sample that is very close to the original motion on the classification boundary. Usually this adversarial sample is in a class that shares the boundary with the class of the original motion. It is possible that these high-connectivity classes share boundaries with many classes so that random attacks are more likely to land in these classes. In addition, the connectivity of classes heavily depends on the classifier itself and that is why different classifiers have different high-connectivity classes. In targeted attack, since our target labels are randomly selected, the confusion is more uniformly distributed, covering all classes.

Rotate right arms backward


Clap above hand

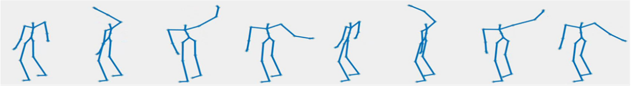
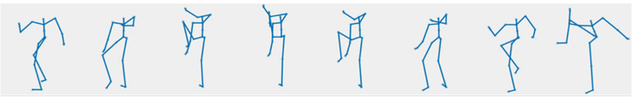
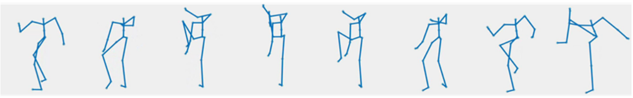

Kick left side

Figure 4. STGCN on HDM05. The ground truth label 'Rotate right arms backward' is misclassified as 'Clap above hand' on untargeted attack, and 'Kick left side' on targeted attack.


Elbow to knee


Sit down

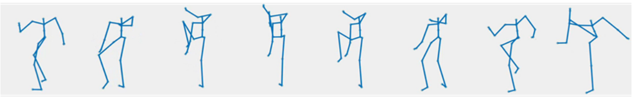
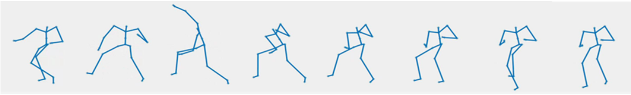
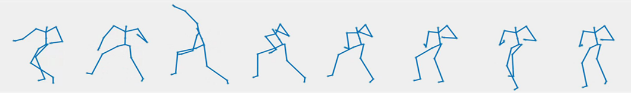

Cartwheel

Figure 5. MSG3D on HDM05. The ground truth label 'Elbow to knee' is misclassified as 'Sit down' on untargeted attack, and 'Cartwheel' on targeted attack.


Punch right front


Punch right side

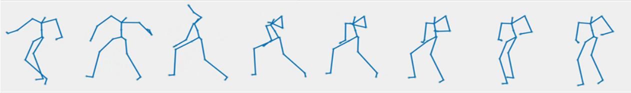
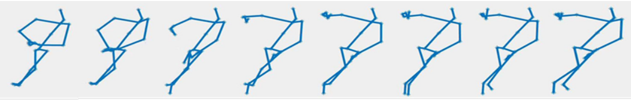
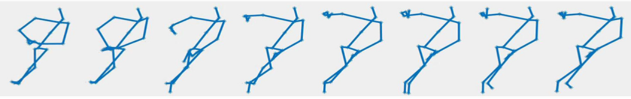

Standing and throw down

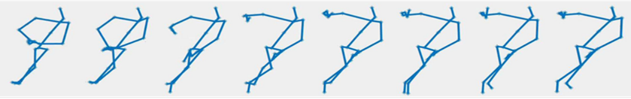
Figure 6. SGN on HDM05. The ground truth label 'Punch right front' is misclassified as 'Punch right side' on untargeted attack, and 'Standing and throw down' on targeted attack.



Taking a selfie

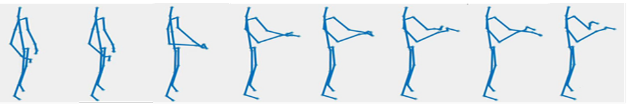


Standing up(from sitting position)

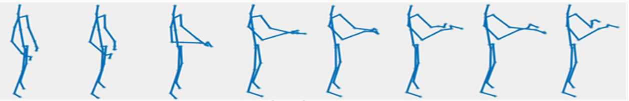


Hand waving

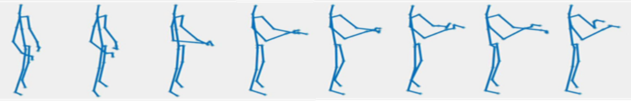
Figure 7. STGCN on NTU. The ground truth label 'Taking a selfie' is misclassified as 'Stand up' on untargeted attack, and 'Hand waving' on targeted attack.



Rub two hands together

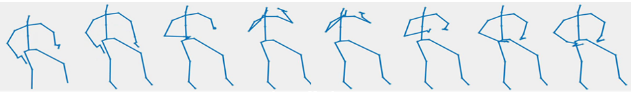


Clapping

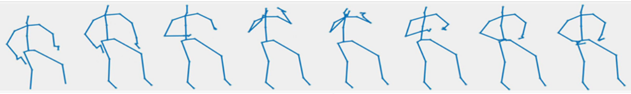


Pick up

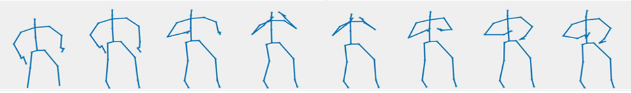
Figure 8. MSG3D on NTU. The ground truth label 'Rub two hands together' is misclassified as 'Clapping' on untargeted attack', and 'Pick up' on targeted attack



Take off glasses



Wear on glasses



Wipe face

Figure 9. SGN on NTU. The ground truth label 'Take off glasses' is misclassified as 'Wear on glasses' on untargeted attack, and 'Wipe face' on targeted attack.

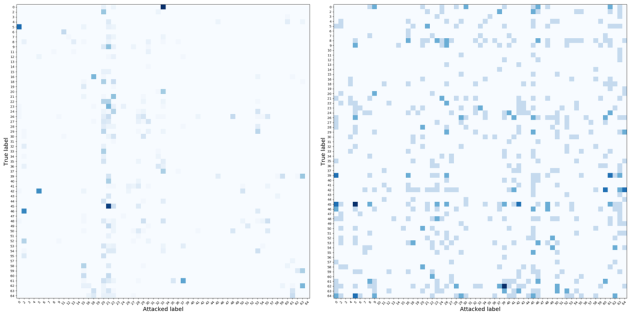


Figure 10. Confusion matrix of STGCN on HDM05. Left is untargeted attack and right is targeted attack. The darker the cell, the higher the value.

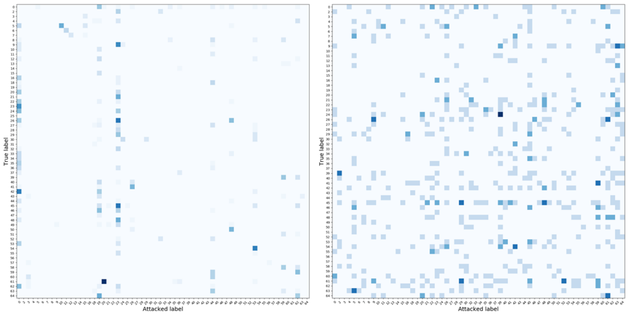


Figure 11. Confusion matrix of MSG3D on HDM05. The left one is untargeted attack and right is targeted attack. The darker the cell, the higher the value.

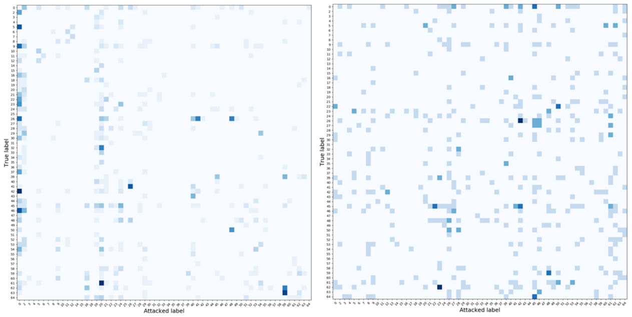


Figure 12. Confusion matrix of SGN on HDM05. The left one is untargeted attack and right is targeted attack. The darker the cell, the higher the value.

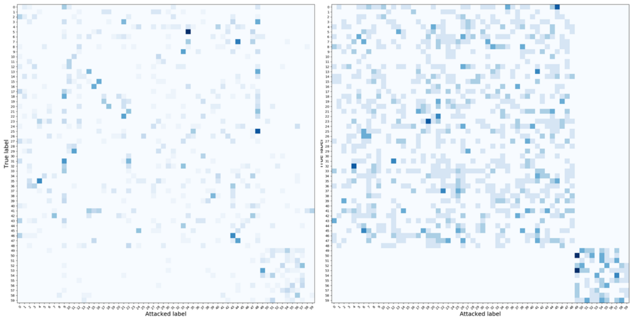


Figure 13. Confusion matrix of STGCN on NTU. The left one is untargeted attack and right is targeted attack. The darker the cell, the higher the value.

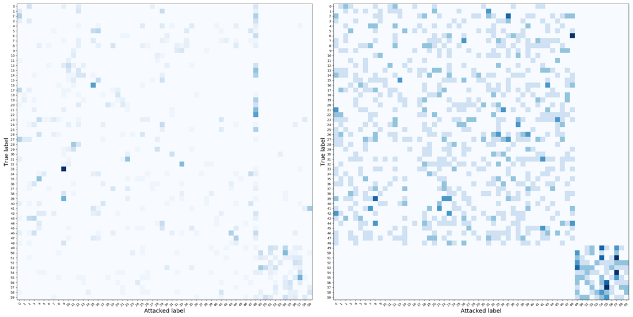


Figure 14. Confusion matrix of MSG3D on NTU. The left one is untargeted attack and right is targeted attack. The darker the cell, the higher the value.

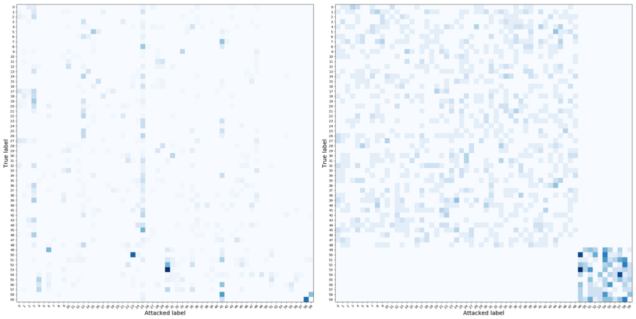


Figure 15. Confusion matrix of SGN on NTU. The left one is untargeted attack and right is targeted attack. The darker the cell, the higher the value.

3. Additional Details of Manifold Projection

The original problem is as follows:

$$\begin{aligned} &\text{minimize} \;\; L(\theta,\theta') + wL(\ddot{\theta},\ddot{\theta}') \\ &\text{subject to} \;\; \theta_i^{\min} \leq \theta'_{\mathbf{i}} \leq \theta_i^{\max}, \\ &C_{\mathbf{x}'} = c \text{ (targeted) or } C_{\mathbf{x}'} \neq C_{\mathbf{x}} \text{ (untargeted).} \end{aligned} \tag{3}$$

where θ and θ' are the joint angles of $\mathbf{x}$ and $\mathbf{x}'$. θ'_i is the i-th joint in every frame of $\mathbf{x}'$ and subject to joint limits bounded by $\theta_i^{\min}$ and $\theta_i^{\max}$. $\ddot{\theta}$ and $\ddot{\theta}'$ are the $2nd$-order derivatives of θ and θ', w is a weight. L is the Euclidean distance. Such a nonlinear optimization problem can be transformed to a barrier problem [1]:

$$\begin{aligned} \min_{\theta'} \;\; L(\theta,\theta') + w_1 L(\ddot{\theta},\ddot{\theta}') + \sum_i^O \mu_i \ln(\theta'_{\mathbf{i}} - \theta_{\mathbf{i}}^{\mathbf{min}}) \\ + \sum_i^O \mu_i \ln(\theta_{\mathbf{i}}^{\mathbf{max}} - \theta'_{\mathbf{i}}) \end{aligned} \tag{4}$$

where μ_i is a barrier parameter. O is the total number of joints in a skeleton. For notational simplicity, we notate $f(\theta') = L(\theta,\theta') + wL(\ddot{\theta},\ddot{\theta}')$. The Karush-Kuhn-Tucker conditions [2] for the barrier problem Eq.5 can be written as:

$$\begin{aligned} \nabla f(\theta') + \sum_i^O \frac{\mu_i}{\theta'_{\mathbf{i}} - \theta_{\mathbf{i}}^{\mathbf{min}}} - \sum_i^O \frac{\mu_i}{\theta_{\mathbf{i}}^{\mathbf{max}} - \theta'_{\mathbf{i}}} &= 0 \\ \mu_i >= 0, \text{ for } i = 1, ..., &O \\ \sum_i^O \mu_i \ln(\theta'_{\mathbf{i}} - \theta_{\mathbf{i}}^{\mathbf{min}}) &= 0 \\ \sum_i^O \mu_i \ln(\theta_{\mathbf{i}}^{\mathbf{max}} - \theta'_{\mathbf{i}}) &= 0 \end{aligned} \tag{5}$$

We apply a damped Newton's method[4] to compute an approximate solution to Eq. 5. More implementation details about the primal-dual interior-point method can be found in [3].

References

[1] Andrew R. Conn, Nicholas I. M. Gould, Dominique Orban, and Philippe L. Toint. A primal-dual trust-region algorithm for non-convex nonlinear programming. *Math. Program.*, 87(2):215–249, 2000. 5

[2] Harold W Kuhn and Albert W Tucker. Nonlinear programming. In *Traces and emergence of nonlinear programming*, pages 247–258. Springer, 2014. 5

[3] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006. 5

[4] Tjalling J Ypma. Historical development of the newton–raphson method. *SIAM review*, 37(4):531–551, 1995. 5