Supplementary Materials for "A Study on Trust Region Update Rules in Newton Methods for Large-scale Linear Classification"

1. Proof of Theorem 1

Proof From (32), for logistic regression,

$$\xi''_{\mathrm{LR}}(\sigma) = \frac{e^{-\sigma}}{(1+e^{-\sigma})^2} > 0, \quad \forall\sigma$$

and for L2 loss, which is not twice differentiable, we use the generalized Hessian with

$$\xi''_{\mathrm{L2}}(\sigma) = \begin{cases} 0, & \text{if } 1 \le \sigma, \\ 2, & \text{if } 1 > \sigma. \end{cases}$$

Thus $\xi''_{\mathrm{L2}}(\sigma) \ge 0, \forall\sigma$. Because $\boldsymbol{s}^k \neq \boldsymbol{0}$, we have

$$g''(\alpha) = (\boldsymbol{s}^k)^T\boldsymbol{s}^k + C\sum\nolimits_{i=1}^{l} \xi''(\sigma)\,(y_i\boldsymbol{x}_i^T\boldsymbol{s}^k)^2 \ge (\boldsymbol{s}^k)^T\boldsymbol{s}^k > 0, \quad \forall\alpha,$$

which implies that $g'(\alpha)$ is strictly increasing. Because $\boldsymbol{s}^k$ is a descent direction satisfying

$$g'(0) = \nabla f(\boldsymbol{w}^k)^T\boldsymbol{s}^k < 0.$$

If $g(\alpha)$ is twice-differentiable (i.e., logistic loss is used), from Taylor expansion of $g(\alpha)$, there exists $\bar{\alpha} \in [0, \alpha]$ such that

$$\begin{aligned} g'(\alpha) &= g'(0) + g''(\bar{\alpha})\alpha \\ &\ge g'(0) + (\boldsymbol{s}^k)^T\boldsymbol{s}^k\alpha \to \infty \text{ as } \alpha \to \infty. \end{aligned} \tag{i}$$

This implies that there exists $\hat{\alpha} > 0$ such that

$$g'(\hat{\alpha}) > 0.$$

With $g'(0) < 0$ and the continuity of $g'(\alpha)$, there exists $\alpha^* \in [0, \hat{\alpha}]$ such that $g'(\alpha^*) = 0$. This root of $g'(\alpha)$ is unique because we have proved earlier that $g'(\alpha), \alpha \ge 0$ is strictly increasing.

For L2-loss SVM, $g(\alpha)$ is not twice differentiable, but we can still obtain (i): from Lemma 3.4 in Mangasarian (2002), $f(\boldsymbol{w})$ of L2-loss SVM is strongly convex with constant 1. Therefore,

$$(\nabla f(\boldsymbol{w}^k + \alpha\boldsymbol{s}^k) - \nabla f(\boldsymbol{w}^k))^T(\alpha\boldsymbol{s}^k) \ge \alpha^2\|\boldsymbol{s}^k\|^2.$$

With

$$g'(\alpha) - g'(0) = (\nabla f(\boldsymbol{w}^k + \alpha\boldsymbol{s}^k) - \nabla f(\boldsymbol{w}^k))^T\boldsymbol{s}^k,$$

(i) follows. ■

2. Other Methods for Solving $\min_\alpha f(\boldsymbol{w}^k + \alpha \boldsymbol{s}^k)$

2.1 Methods Based on First- or Second-order Information

If we consider gradient descent or Newton methods to minimize (29), the search direction at any α is

$$\Delta\alpha = -g'(\alpha) \quad \text{or} \quad -\frac{g'(\alpha)}{g''(\alpha)}, \tag{ii}$$

which, from the discussion in (30)-(33), can be obtained by $O(l)$ operations. For the convergence, we can for example consider backtrack line search by finding a step size r such that

$$g(\alpha + r\Delta\alpha) \le g(\alpha) + \tau r g'(\alpha)\Delta\alpha,$$

where $\tau \in (0, 1)$. For each r tested, $O(l)$ cost is needed for calculating $g(\alpha+r\Delta\alpha)$. Therefore, the total cost is

$$O(l) \times (\#\text{iterations}) \times (1 + \text{average } \#\text{line-search steps per iteration}). \tag{iii}$$

Note that the #iterations here means the number of iterations to minimize (29) rather than that for the linear-classification problem in (3). From the analysis in (iii), we may not prefer using regular optimization methods because of the following concerns.

1. For a simple one-variable problem, a method that needs line search is rather complicated.

2. In contrast to the multi-variable scenarios of solving (3), where from (11) the cost of line search is negligible, here it becomes dominant.

3. While we aim to obtain an α_k^* that better minimizes $f(\boldsymbol{w}^k + \alpha\boldsymbol{s}^k)$, there is no need to very accurately conduct the minimization. The reason is that α_k^* is used only to adjust the size of the trust region. However, methods based on first- or second- order information aim to accurately minimize a problem rather than finding an approximate solution.

Therefore, we choose to design some simple yet effective approaches like the bisection method in Section 3.3.1 of the main paper and the cubic interpolation in the next subsection.

2.2 Cubic Interpolation

In Section 2.2 of the paper, α_k^* in the rule (20) is derived by minimizing a quadratic interpolation of $f(\boldsymbol{w}^k + \alpha\boldsymbol{s}^k)$. Three values $g(0)$, $g'(0)$ and $g(1)$ were used. We can consider a more accurate interpolation by using, for example, $g'(1)$, $g''(0)$ or $g''(1)$. We can afford to calculate these values because as indicated in (30)-(32), each of them takes only $O(l)$ operations. However, the polynomial obtained for interpolation may have a higher degree and be non-convex. The minimization becomes difficult. Here we investigate the use of a degree-3 polynomial interpolation. A function

$$\phi_c(\alpha) = a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0, \tag{iv}$$

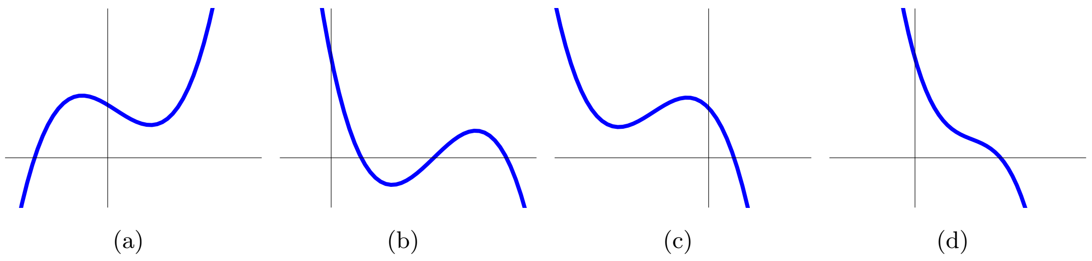


Figure 1: Possible types of curves of $\phi_c(\alpha)$

is constructed to satisfy

$$\begin{aligned} \phi_c(0) &= f(\boldsymbol{w}^k), & \phi_c'(0) &= \nabla f(\boldsymbol{w}^k)^T \boldsymbol{s}^k, \\ \phi_c(1) &= f(\boldsymbol{w}^k + \boldsymbol{s}^k), & \phi_c'(1) &= \nabla f(\boldsymbol{w}^k + \boldsymbol{s}^k)^T \boldsymbol{s}^k. \end{aligned} \tag{v}$$

We leave details in Section 2.2.1.

Our experiments indicate that this approach performs well when the logistic loss is used. However, if the squared loss (i.e., least square regression) is considered, $f(\boldsymbol{w}^k + \alpha\boldsymbol{s}^k)$ is a quadratic function of α and therefore $\alpha_3 = 0$. In the formulation to get α_k^*, a 0/0 operation occurs. Therefore, we must specially handle the situation when $\alpha_3 \approx 0$. This issue and the non-convexity of $\phi_c(\alpha)$ in (iv) may make this approach a less ideal choice than the bisection method.

2.2.1 Details of Obtaining α_k^* by Cubic Interpolation

From (v) we can easily derive that

$$\begin{aligned} a_0 &= \phi_c(0), & a_2 &= 3(\phi_c(1) - \phi_c(0)) - 2\phi_c'(0) - \phi_c'(1), \\ a_1 &= \phi_c'(0), & a_3 &= \phi_c'(1) + \phi_c'(0) - 2(\phi_c(1) - \phi_c(0)). \end{aligned}$$

Because $\phi_c'(0) = \nabla f(\boldsymbol{w}^k)^T \boldsymbol{s}^k < 0$, we have that $\phi_c(\alpha)$ is like one of the curves in Figure 1.

Though $\phi_c(\alpha)$ is not convex, from Figures 1(a) and 1(b), and the fact that we have used information at $\alpha = 0$ and 1 in (v), we may consider the positive local minimum of $\phi_c(\alpha)$ as α_k^* to approximate the minimum of (29) by solving

$$\phi_c'(\alpha) = 3a_3\alpha^2 + 2a_2\alpha + a_1 = 0. \tag{vi}$$

For the situations like Figures 1(c) and 1(d), $\phi_c(\alpha)$ has no positive local minimum. However, we can directly use $\alpha_k^* = +\infty$. The reason is that α_k^* is used only as additional information in updating the trust-region size.

Next we discuss details of obtaining α_k^*. Because $\boldsymbol{s}^k$ is a descent direction,

$$a_1 = \phi_c'(0) = \nabla f(\boldsymbol{w}^k)^T \boldsymbol{s}^k < 0. \tag{vii}$$

If $a_3 \neq 0$, we can solve (vi) by discussing the roots of the quadratic equation. Firstly, (vi) has real roots only if

$$a_2^2 - 3a_3a_1 \geq 0. \tag{viii}$$

Then the roots of (vi) are

$$\alpha_1 = \frac{-a_2 + \sqrt{a_2^2 - 3a_3a_1}}{3a_3} \quad \text{and} \quad \alpha_2 = \frac{-a_2 - \sqrt{a_2^2 - 3a_3a_1}}{3a_3}.$$

We then consider the following situations:

1. $a_3 > 0$.
   With $a_1 < 0$ from (vii),
   $$a_2^2 - 3a_3a_1 > a_2^2 \geq 0.$$
   Thus
   $$\alpha_2 < 0 < \alpha_1,$$
   and because
   $$\begin{aligned}\phi_c''(\alpha_1) &= 6a_3\alpha_1 + 2a_2 \\ &= 2\sqrt{a_2^2 - 3a_3a_1} > 0,\end{aligned} \tag{ix}$$
   α_1 is a local minimum of $\phi_c(\alpha)$. Then we have
   $$\alpha_k^* = \alpha_1.$$
   This case corresponds to Figure 1(a).

2. $a_3 < 0$ and $a_2 > \sqrt{3a_3a_1}$.
   With $a_1 < 0$, we have
   $$3a_3a_1 > 0.$$
   Thus
   $$a_2 > \sqrt{3a_3a_1} > 0. \tag{x}$$
   We have
   $$0 < \alpha_1 < \alpha_2.$$
   From (ix) and (x),
   $$\phi_c''(\alpha_1) > 0,$$
   so α_1 is a local minimum. Thus we choose
   $$\alpha_k^* = \alpha_1.$$
   This case corresponds to Figure 1(b).

3. $a_3 < 0$ and $a_2 < -\sqrt{3a_3a_1}$.
   We have
   $$\begin{aligned}\phi_c'(\alpha) &= 3a_3\alpha^2 + 2a_2\alpha + a_1 \\ &\leq a_1 < 0, \quad \forall \alpha \geq 0.\end{aligned}$$
   Thus $\phi_c(\alpha), \alpha \geq 0$ is a strictly decreasing function and we set
   $$\alpha_k^* = +\infty.$$
   This case corresponds to Figure 1(c).

4. $a_3 < 0$ and $a_2^2 - 3a_3a_1 \leq 0$. We have

$$\phi_c'(\alpha) = 3a_3(\alpha + \frac{a_2}{3a_3})^2 + a_1 - \frac{a_2^2}{3a_3}$$
$$\begin{cases} < a_1 - \dfrac{a_2^2}{3a_3} \leq 0, & \text{if } \alpha \neq \dfrac{-a_2}{3a_3}, \\ \leq a_1 - \dfrac{a_2^2}{3a_3} \leq 0, & \text{if } \alpha = \dfrac{-a_2}{3a_3}. \end{cases}$$

   Therefore, $\phi_c(\alpha)$ is a strictly decreasing function in $\alpha \in (-\infty, -a_2/3a_3]$ and $[-a_2/3a_3, \infty)$. Together, $\phi_c(\alpha)$ is strictly decreasing in $\alpha \in \mathbb{R}$. Thus $\phi_c(\alpha)$ has no positive local minimum and hence we set

$$\alpha_k^* = +\infty.$$

   This case corresponds to Figure 1(d).

In summary, we have

$$\alpha_k^* = \begin{cases} \dfrac{-a_2 + \sqrt{a_2^2 - 3a_3a_1}}{3a_3}, & \text{if } a_3 > 0 \text{ or } (a_3 < 0 \text{ and } a_2 > \sqrt{3a_3a_1}), \\ +\infty, & \text{otherwise.} \end{cases}$$

If $a_3 = 0$, then $\phi_c(\alpha)$ is a quadratic function. We can then go back to use the quadratic interpolation in (20).

3. Data Statistics

Table 1 shows the statistics of each data set used in our experiments.

4. Analysis of the Running Time of Bisection and CG Procedures

Table 2 investigates if the accurate minimization of $f(\boldsymbol{w}^k + \alpha \boldsymbol{s}^k)$ by the bisection method causes a significant increase of the running time. We begin with checking the average cost ratio between one step in the bisection method and one CG step. From (iii) and (11), their costs are respectively $O(l)$ and $O(\#\text{nnz})$, where #nnz if the total number of non-zero feature values in the l training instances. Clearly the ratio is generally small. We then present the total number of bisection steps and the total number of CG steps in the whole training process. The number of CG steps is always higher. Finally we show the ratio of total time. Results confirm that the total efforts in finding α_k^* is much smaller than CG procedures for finding directions.

5. Complete Results on Logistic Regression

We present the following figures.

- Figures 2-6 : the evaluation of the proposed changes on the trust-region update rule. We show $C = \{0.01, 0.1, 1, 10, 100\} \times C_{\text{best}}$. Figure 3 in the paper is a subset of the results presented here.

Table 1: Data statistics. The density is the average number of non-zeros per instance. C_{best} is the regularization parameter selected by cross validation.

Data sets	#instances	#features	density	$\log_2(C_{\text{best}})$	
				LR	L2
`HIGGS`	11,000,000	28	92.1057%	-6	-12
`w8a`	49,749	300	3.8834%	8	2
`rcv1`	20,242	47,236	0.1568%	6	0
`real-sim`	72,309	20,958	0.2448%	3	-1
`news20`	19,996	1,355,191	0.0336%	9	3
`url`	2,396,130	3,231,962	0.0036%	-7	-10
`yahoojp`	176,203	832,026	0.0160%	3	-1
`yahookr`	460,554	3,052,939	0.0111%	6	1
`webspam`	350,000	16,609,143	0.0220%	2	-3
`kdda`	8,407,752	20,216,831	0.0001%	-3	-5
`kddb`	19,264,097	29,890,095	0.0001%	-1	-4
`criteo`	45,840,617	1,000,000	0.0039%	-15	-12
`kdd12`	149,639,105	54,686,452	0.00002%	-4	-11

Table 2: A comparison on running time of the bisection method to minimize $f(\boldsymbol{w}^k + \alpha \boldsymbol{s}^k)$ and the CG procedure. We use $C = C_{\text{best}}$ and LIBLINEAR's default stopping condition.

Data sets	Avg. time per step	Avg. # bisection steps per search	# total steps		Total time
	Bisection/CG		Bisection	CG	Bisection/CG
`HIGGS`	30.15 %	5.00	21	50	12.66 %
`w8a`	66.81 %	7.00	64	74	57.78 %
`rcv1`	6.35 %	6.83	37	40	5.87 %
`real-sim`	12.16 %	6.25	44	47	11.39 %
`news20`	0.61 %	6.38	35	43	0.50 %
`url`	4.61 %	5.85	46	57	3.72 %
`yahoojp`	3.32 %	6.00	42	80	1.74 %
`yahookr`	1.31 %	6.73	63	371	0.22 %
`webspam`	0.07 %	7.00	46	77	0.04 %
`kdda`	12.57 %	7.00	70	872	1.01 %
`kddb`	15.86 %	7.00	63	1279	0.78 %

- Figures 7-11: the comparison between exact and backtrack line search methods. We show $C = \{0.01, 0.1, 1, 10, 100\} \times C_{\text{best}}$. Figure 4 in the paper is a subset of the results presented here.

- Figures 12-16: the comparison between line search and trust region methods. We show $C = \{0.01, 0.1, 1, 10, 100\} \times C_{\text{best}}$. Figure 5 in the paper is a subset of the results presented here.

6. Results for L2-loss SVM

We show the same set of results as those in Section 5, and have the following observations.

- The current trust-region update rule in LIBLINEAR also has slow convergence in some situations.
- Line search performs well because unlike the situation in logistic regression it suffers less from slow convergence when C is large. The reason might be that the L2 loss function is close to a quadratic function. Then the resulting Newton direction more accurately minimizes the function value and the line search procedure often succeeds at $\alpha_k = 1$.
- The new trust-region update rule (28) is effective. Now trust region methods can compete with line search methods for L2-loss SVM.
- For the rule (35) without using $\alpha_k^*\|\boldsymbol{s}^k\|$, the performance is generally competitive. However, for few situations such as Figures 29(b) and 29(c), it is worse than the update rule (28).

7. Discussion of Related Works

The work (Walmag and Delhez, 2005) addresses the situation when ρ is very large (i.e., actual reduction much bigger than predicted reduction). They think Δ_k should remain about the same in such a situation. It will be interesting to try their setting, though currently we do not anticipate significant improvements because we have incorporated $\alpha_k^*\|\boldsymbol{s}^k\|$ in the update rule. Note that from the discussion in this work, $\alpha_k^*\|\boldsymbol{s}^k\|$ may help to quickly adjust Δ_k. Thus even if a too large ρ causes an inappropriate Δ_k, the problem may be quickly fixed.

In Fan and Yuan (2001), the authors propose a rule of

$$\Delta_{k+1} = \alpha_{k+1}^*\|\nabla f(\boldsymbol{w}^k)\|, \tag{xi}$$

where

$$\alpha_{k+1}^* = \begin{cases} \cdots \\ (\text{a value} > 1) \cdot \alpha_k^*, & \text{if } \rho_k \geq 0.25 \text{ and } \|\boldsymbol{s}^k\| > \dfrac{\Delta_k}{2}. \end{cases}$$

Their use of (xi) and $\|\boldsymbol{s}^k\| > \Delta_k/2$ are slightly related to our settings though the two rules still differ in many aspects. Ours obtains α_k^* by approximately minimizing $f(\boldsymbol{w}^k + \alpha\boldsymbol{s}^k)$, and we use $\|\boldsymbol{s}^k\|$ rather than $\|\nabla f(\boldsymbol{w}^k)\|$. Further, we embed $\alpha_k^*\|\boldsymbol{s}^k\|$ in a standard trust-region update rule but theirs does not.

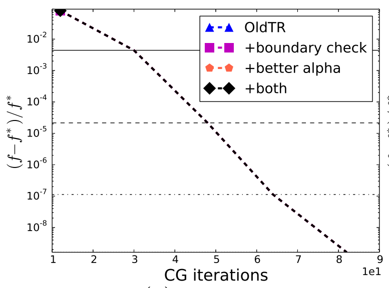
References

Jin-yan Fan and Ya-xiang Yuan. A new trust region algorithm with trust region radius converging to zero. In *Proceeding of the 5th International Conference on Optimization: Techiniques and Applications*, pages 786–794, 2001.

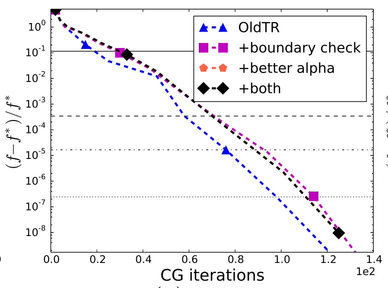
Olvi L. Mangasarian. A finite Newton method for classification. *Optimization Methods and Software*, 17(5):913–929, 2002.

Jérôme MB Walmag and Éric JM Delhez. A note on trust-region radius update. *SIAM Journal on Optimization*, 16(2):548–562, 2005.

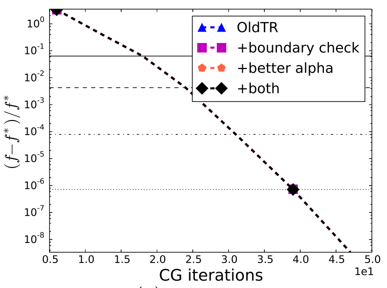
Po-Wei Wang, Ching-Pei Lee, and Chih-Jen Lin. The common directions method for regularized empirical loss minimization. Technical report, National Taiwan University, 2016.



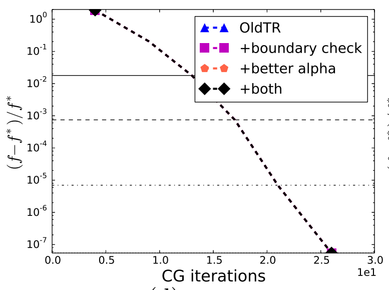
(a) `HIGGS`



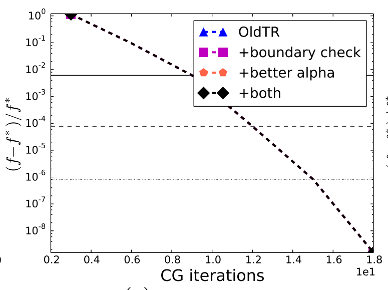
(b) `w8a`



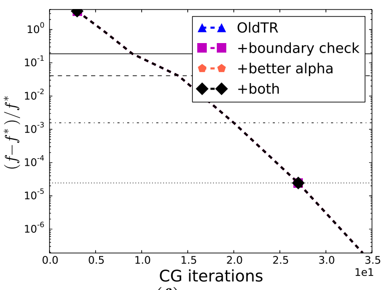
(c) `news20`



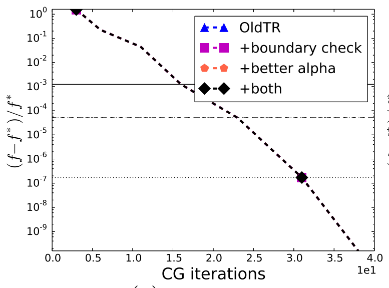
(d) `rcv1`



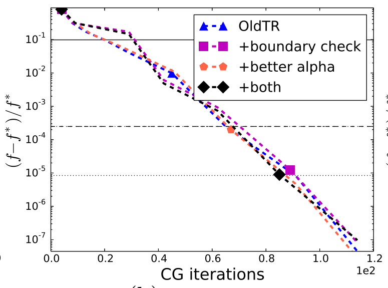
(e) `real-sim`



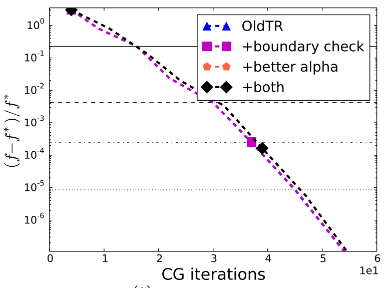
(f) `url`



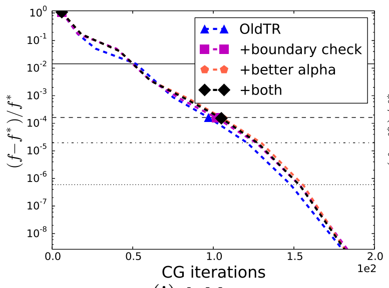
(g) `yahoojp`



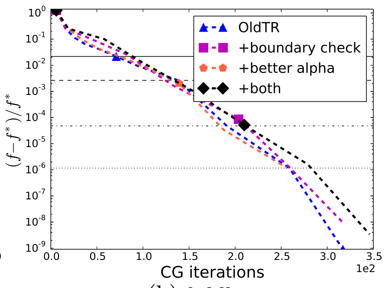
(h) `yahookr`



(i) `webspam`



(j) `kdda`



(k) `kddb`

Figure 2: Convergence of using different trust region update rules. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = 0.01C_{\text{best}}$.

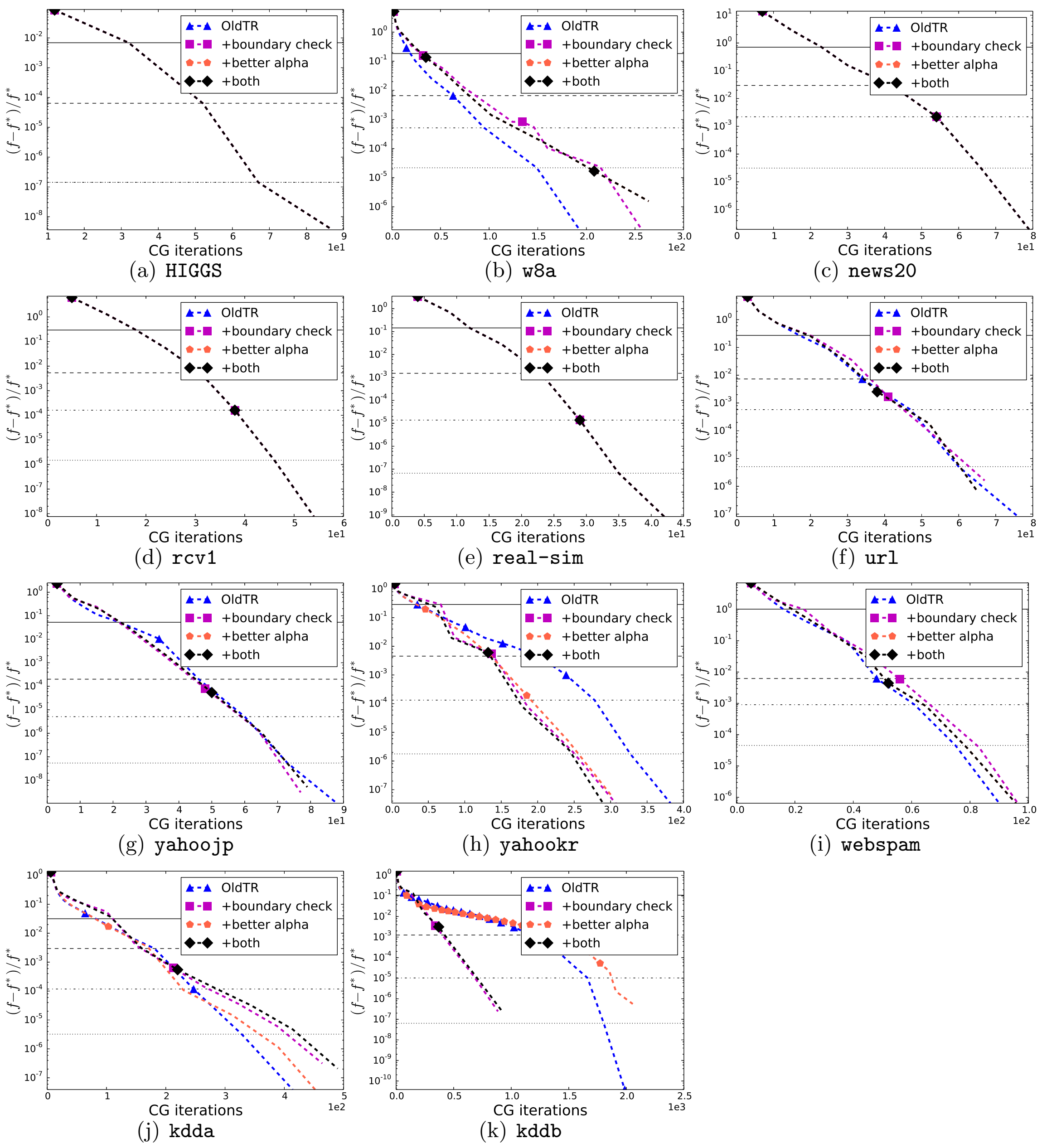
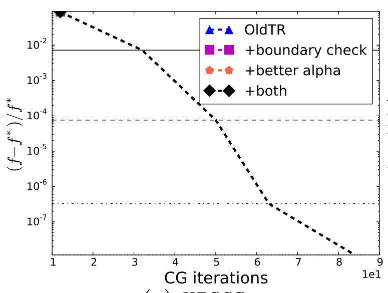
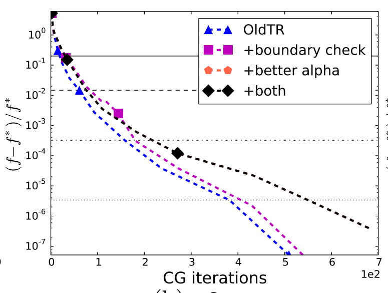


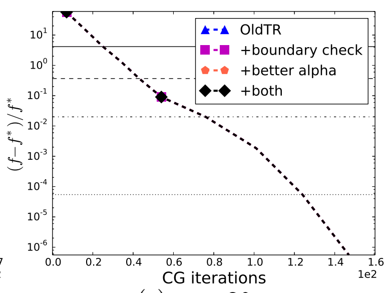
Figure 3: Convergence of using different trust region update rules. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = 0.1C_{\text{best}}$.



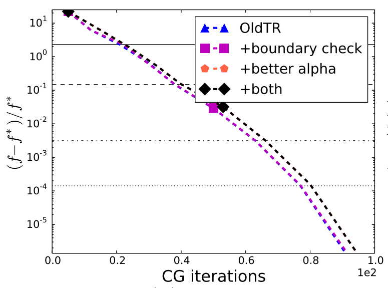
(a) HIGGS



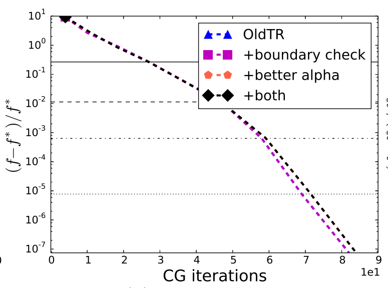
(b) w8a



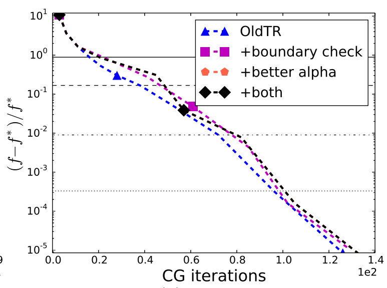
(c) news20



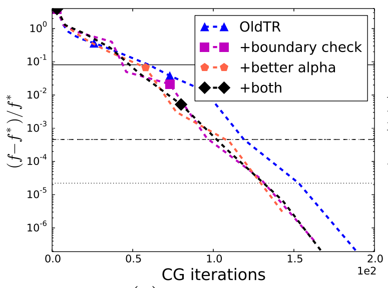
(d) rcv1



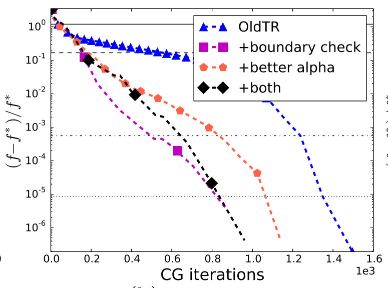
(e) real-sim



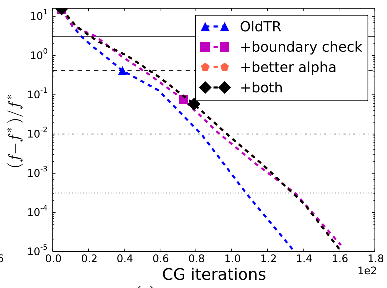
(f) url



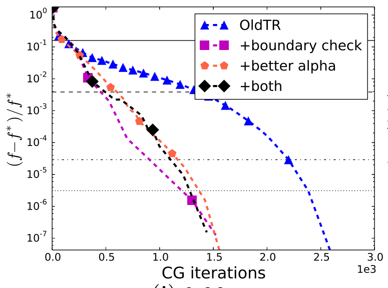
(g) yahoojp



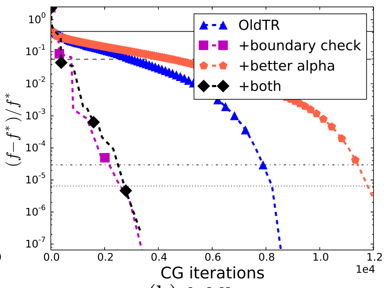
(h) yahookr



(i) webspam



(j) kdda



(k) kddb

Figure 4: Convergence of using different trust region update rules. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = C_{\text{best}}$.

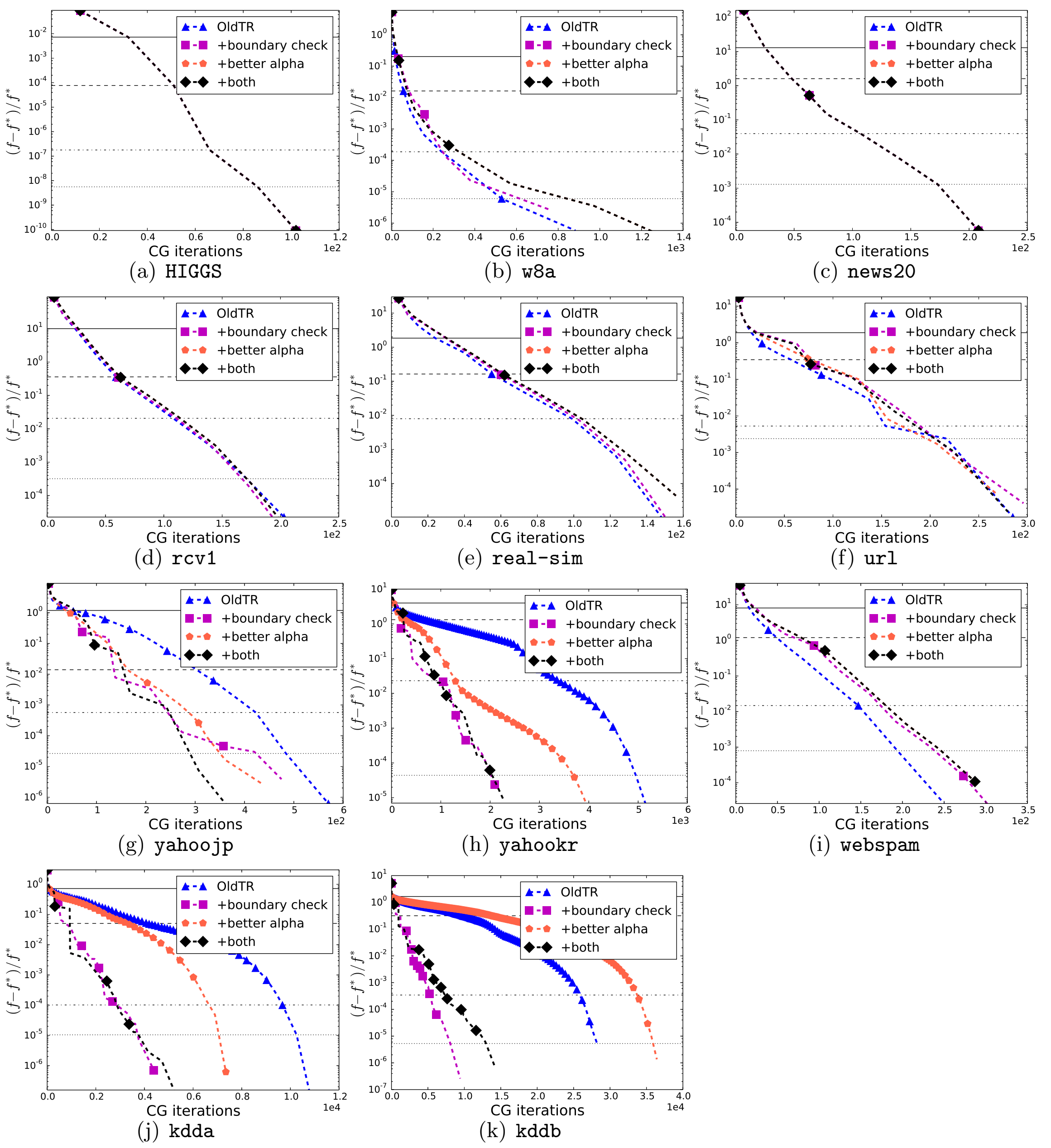
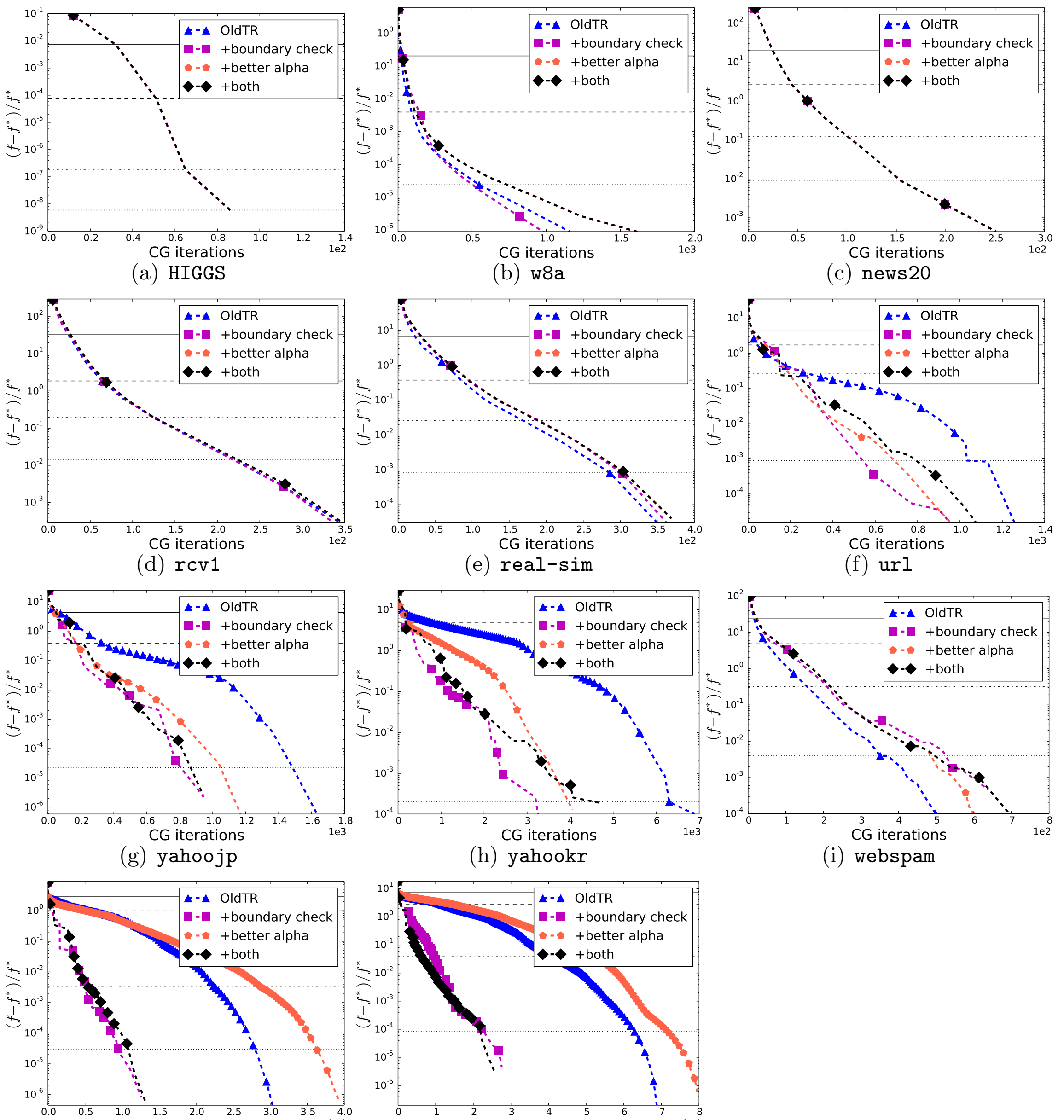


Figure 5: Convergence of using different trust region update rules. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = 10C_{\text{best}}$.



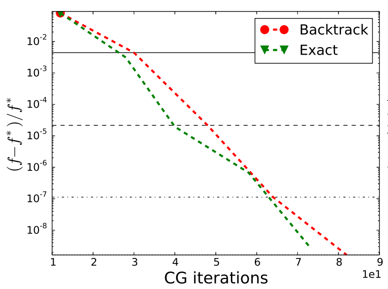
(a) HIGGS (b) w8a (c) news20

(d) rcv1 (e) real-sim (f) url

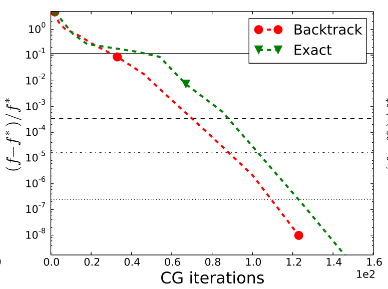
(g) yahoojp (h) yahookr (i) webspam

(j) kdda (k) kddb

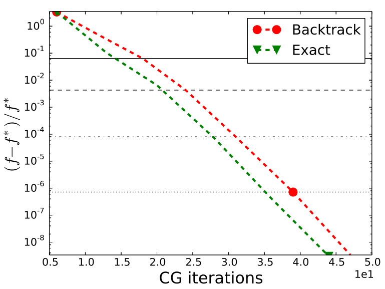
Figure 6: Convergence of using different trust region update rules. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = 100C_{\text{best}}$.



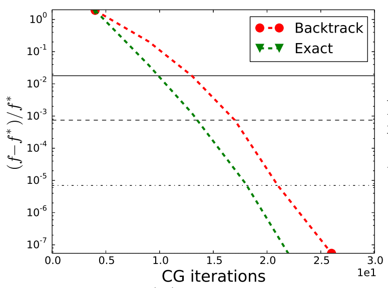
(a) HIGGS



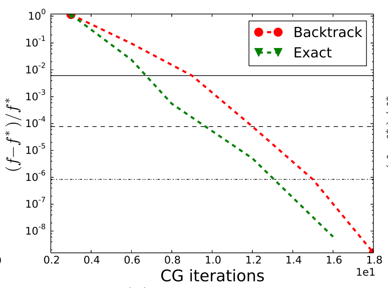
(b) w8a



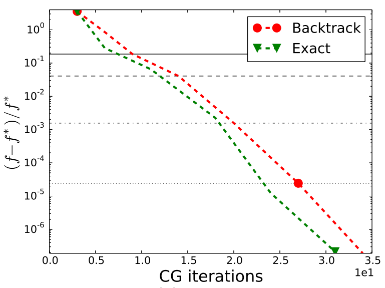
(c) news20



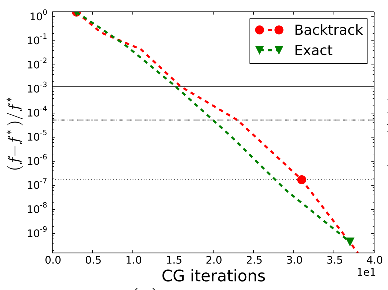
(d) rcv1



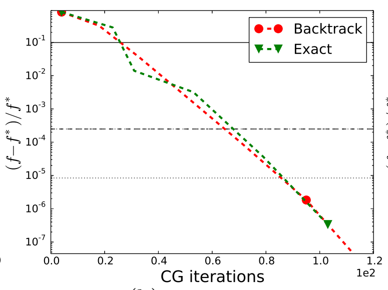
(e) real-sim



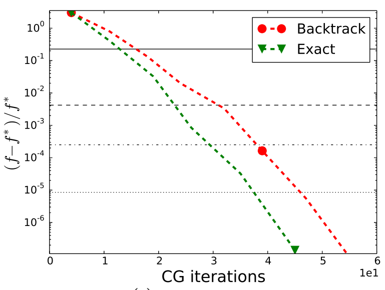
(f) url



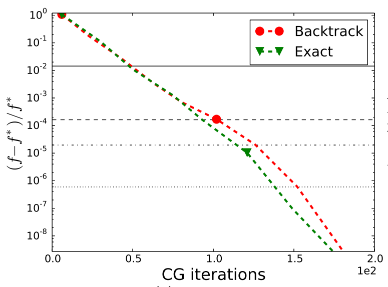
(g) yahoojp



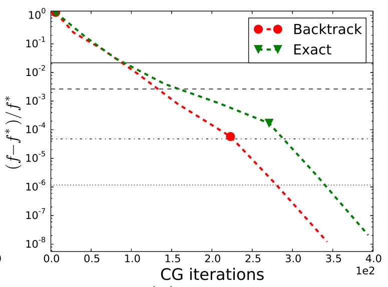
(h) yahookr



(i) webspam

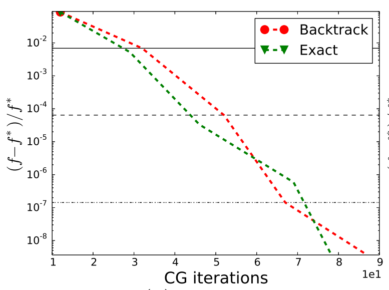


(j) kdda

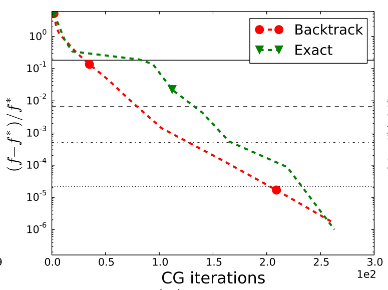


(k) kddb

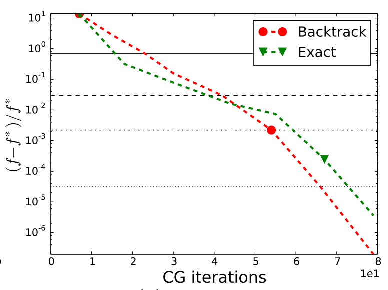
Figure 7: Comparisons of line search and exact line The x-axis shows the search. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = 0.01C_{\text{best}}$.



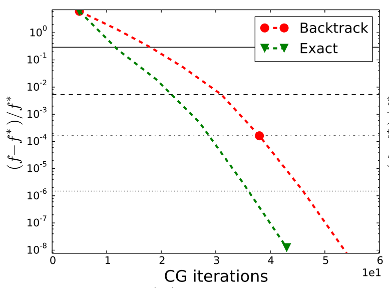
(a) `HIGGS`



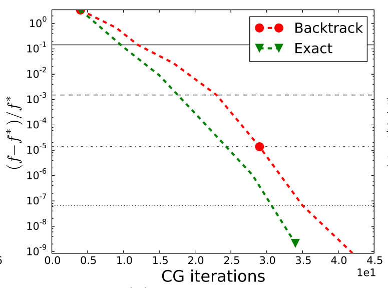
(b) `w8a`



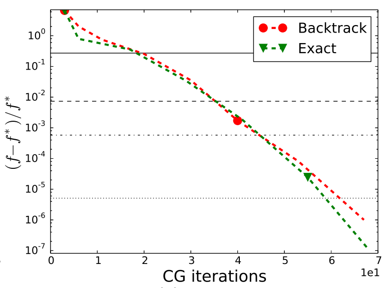
(c) `news20`



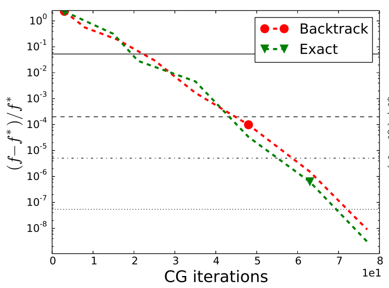
(d) `rcv1`



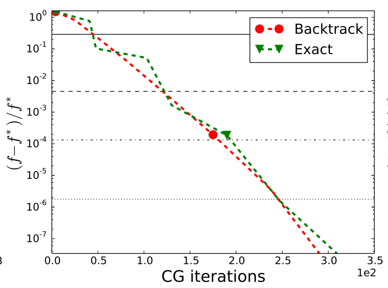
(e) `real-sim`



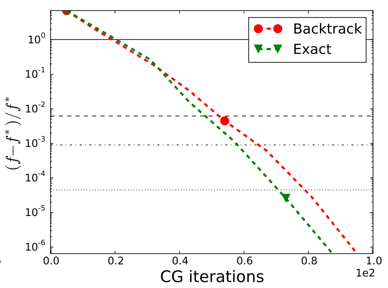
(f) `url`



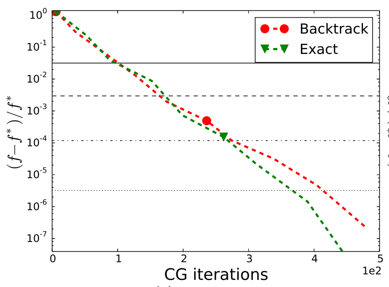
(g) `yahoojp`



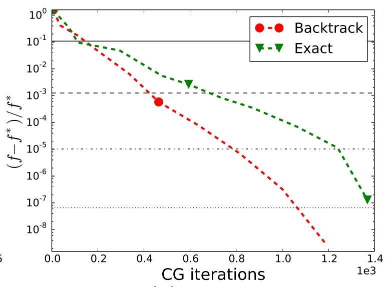
(h) `yahookr`



(i) `webspam`



(j) `kdda`



(k) `kddb`

Figure 8: Comparisons of line search and exact line The x-axis shows the search. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = 0.1C_{\text{best}}$.

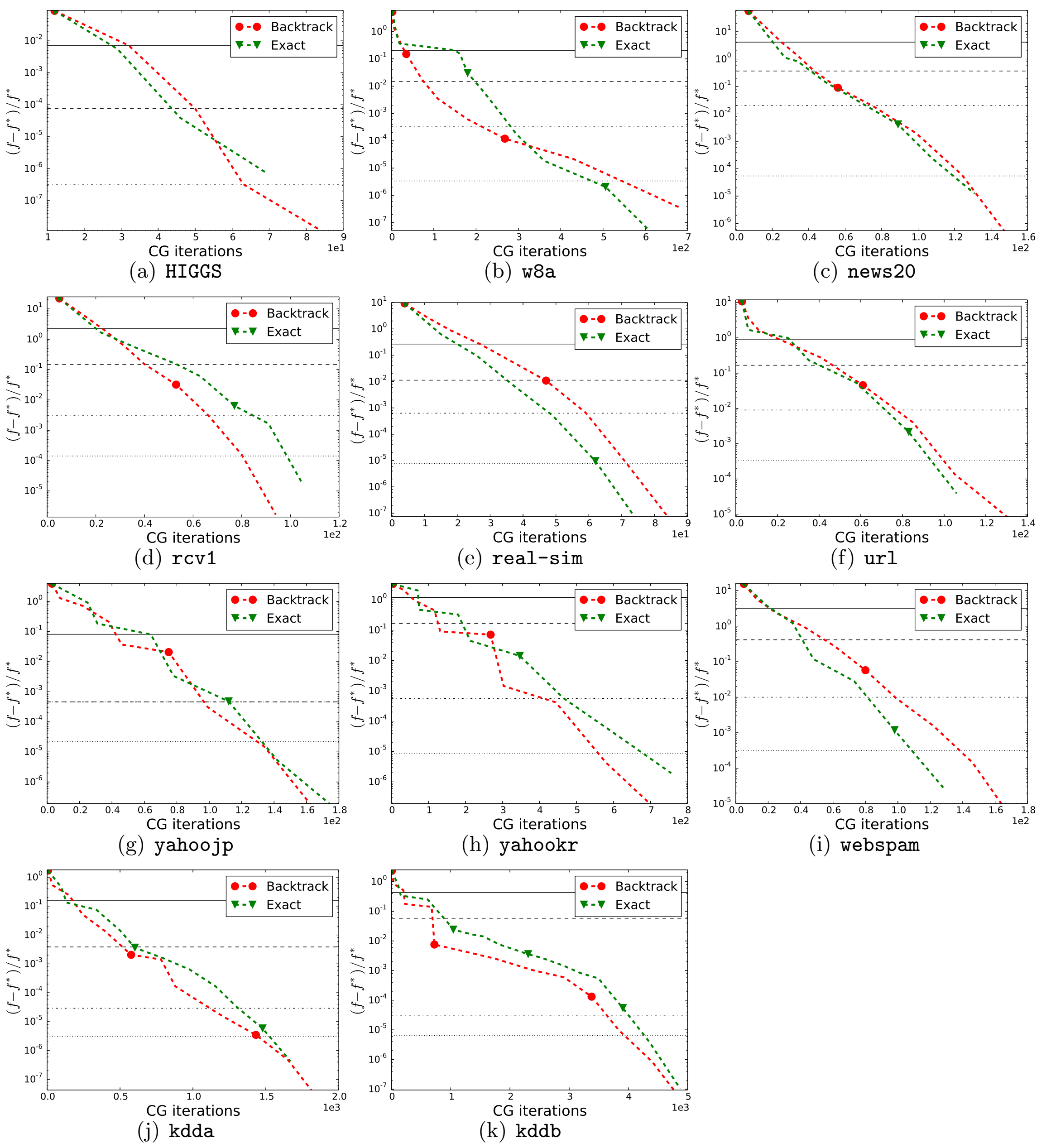
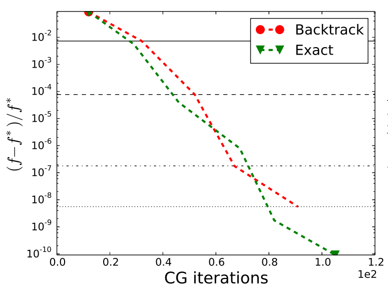
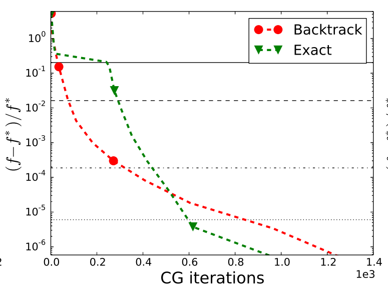


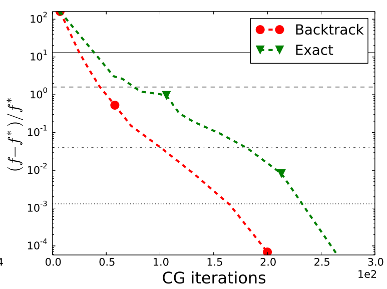
Figure 9: Comparisons of line search and exact line The x-axis shows the search. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = C_{\text{best}}$.



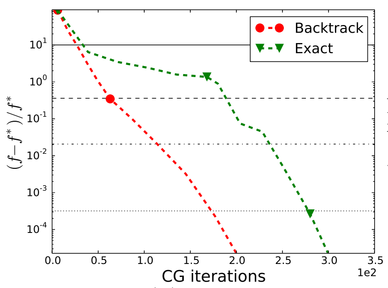
(a) HIGGS



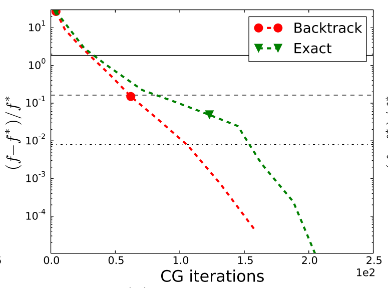
(b) w8a



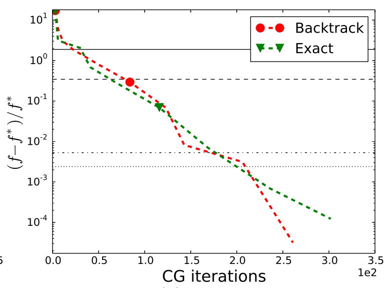
(c) news20



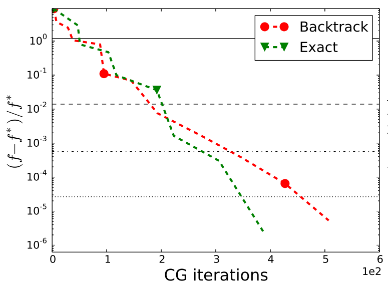
(d) rcv1



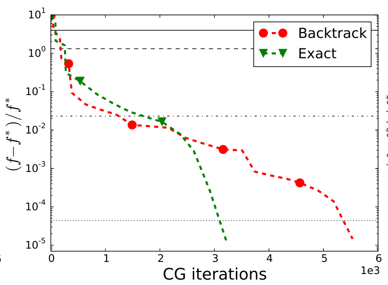
(e) real-sim



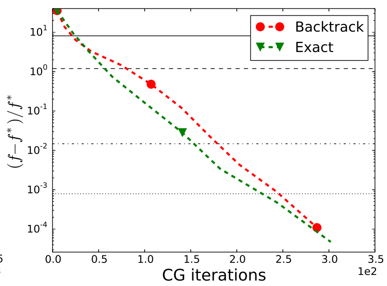
(f) url



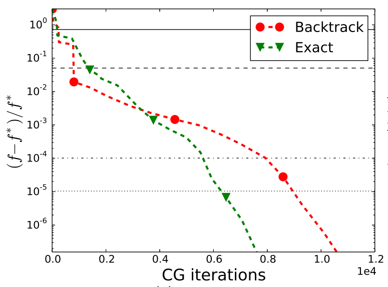
(g) yahoojp



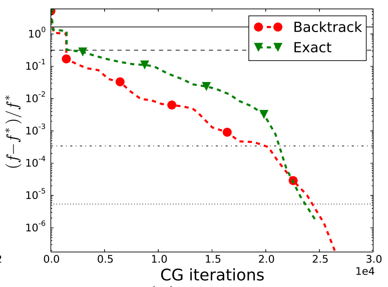
(h) yahookr



(i) webspam

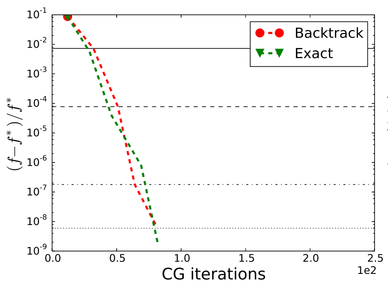


(j) kdda

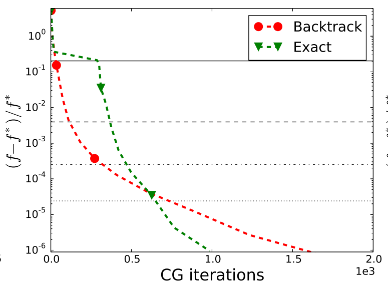


(k) kddb

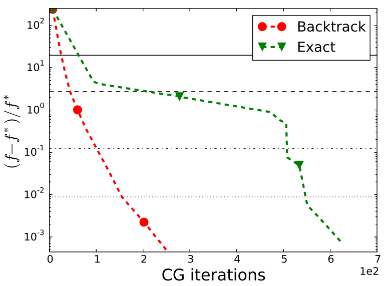
Figure 10: Comparisons of line search and exact line The x-axis shows the search. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = 10C_{\text{best}}$.



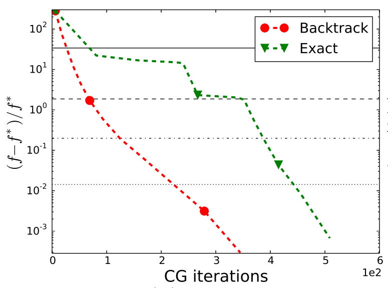
(a) `HIGGS`



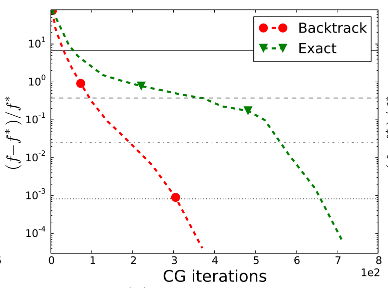
(b) `w8a`



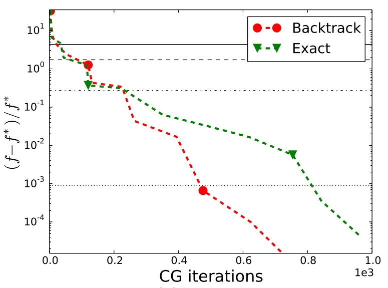
(c) `news20`



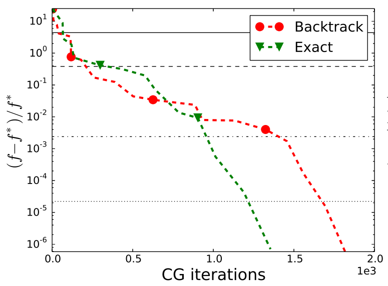
(d) `rcv1`



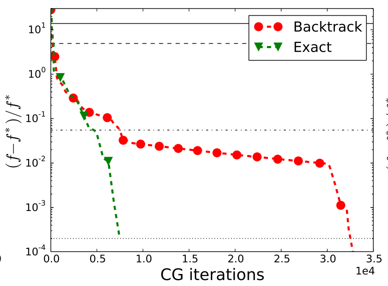
(e) `real-sim`



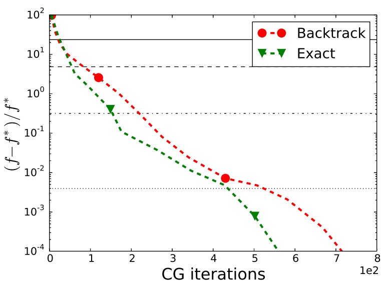
(f) `url`



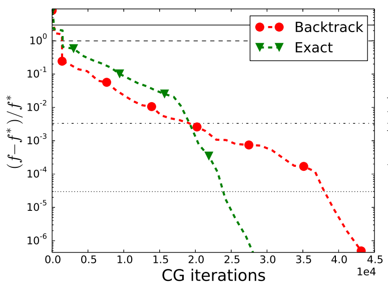
(g) `yahoojp`



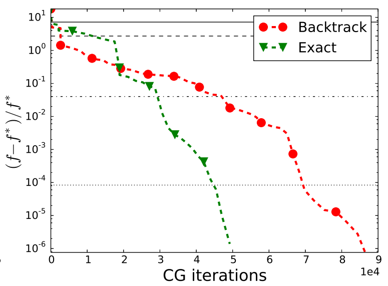
(h) `yahookr`



(i) `webspam`

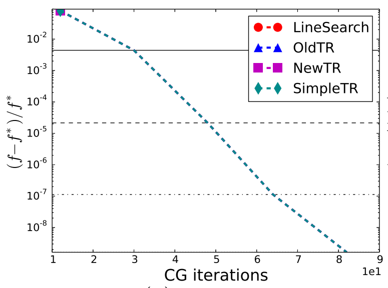


(j) `kdda`

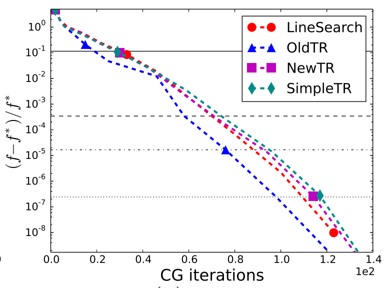


(k) `kddb`

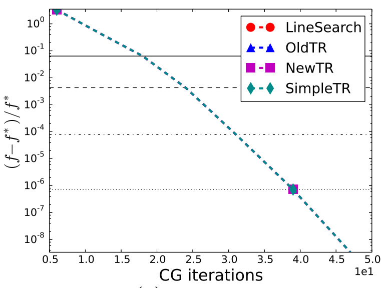
Figure 11: Comparisons of line search and exact line The x-axis shows the search. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = 100C_{\text{best}}$.



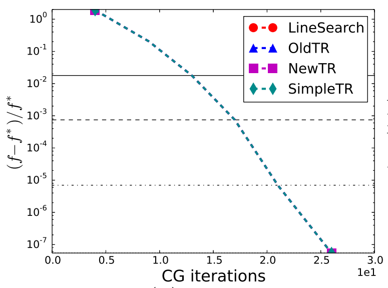
(a) HIGGS



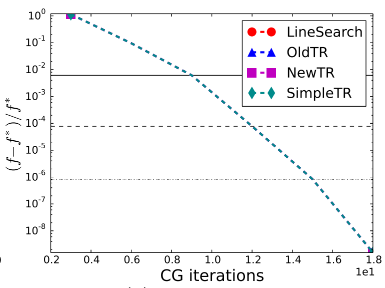
(b) w8a



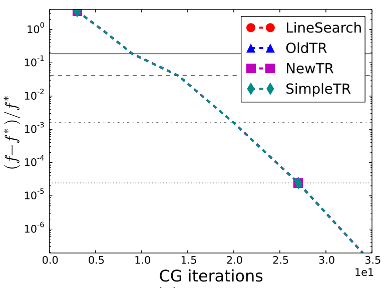
(c) news20



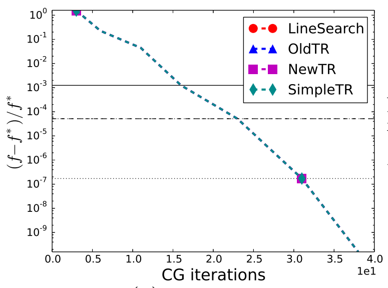
(d) rcv1



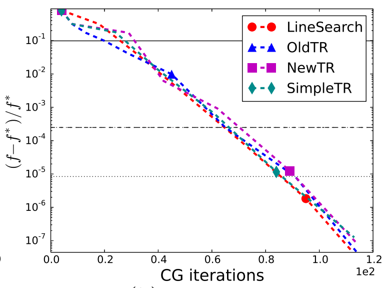
(e) real-sim



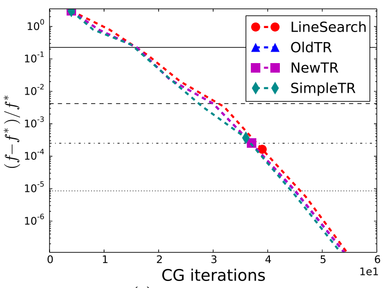
(f) url



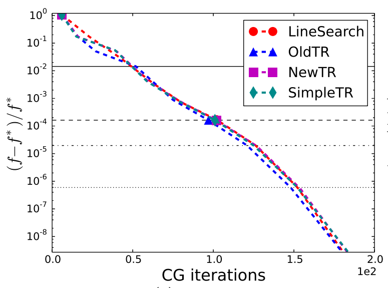
(g) yahoojp



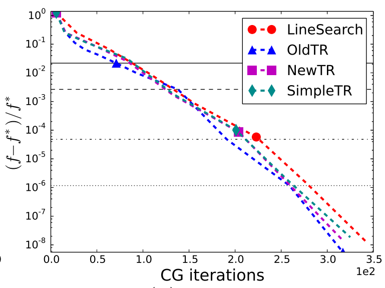
(h) yahookr



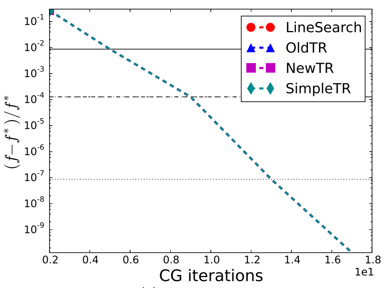
(i) webspam



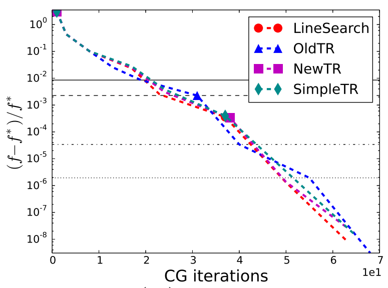
(j) kdda



(k) kddb

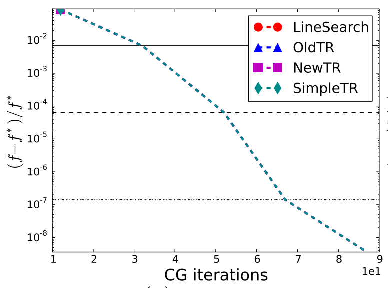


(l) criteo

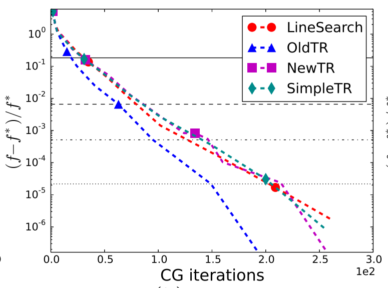


(m) kdd12

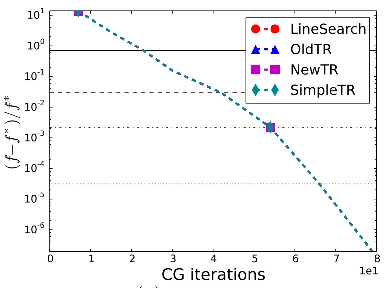
Figure 12: Comparison of line search and different trust region The x-axis shows the methods. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = 0.01C_{\text{best}}$.



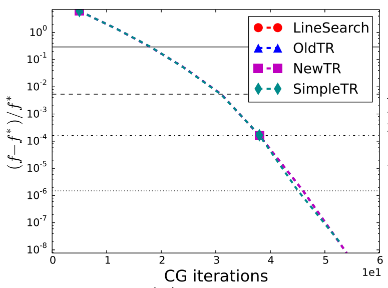
(a) `HIGGS`



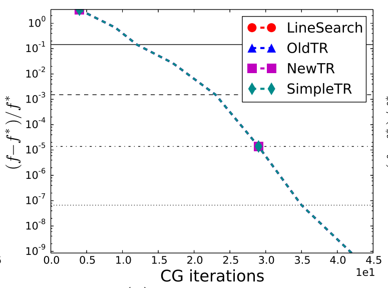
(b) `w8a`



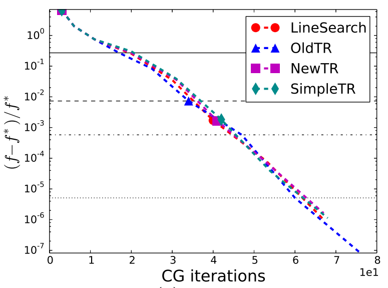
(c) `news20`



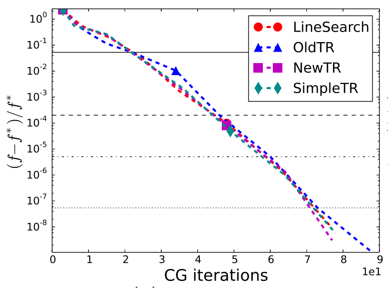
(d) `rcv1`



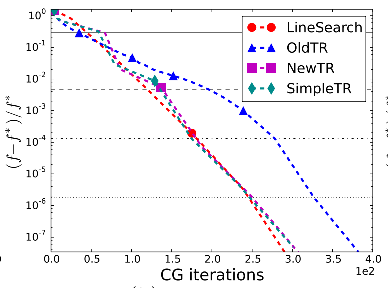
(e) `real-sim`



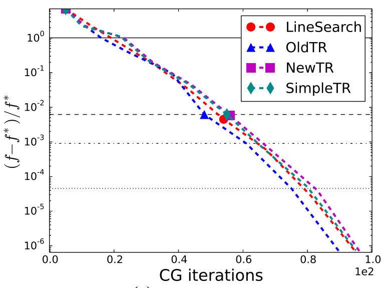
(f) `url`



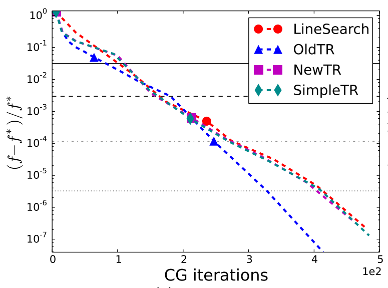
(g) `yahoojp`



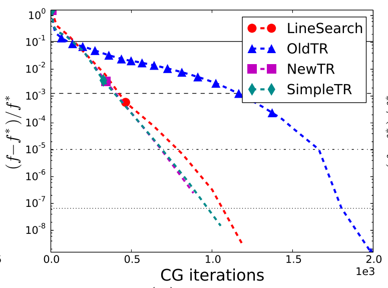
(h) `yahookr`



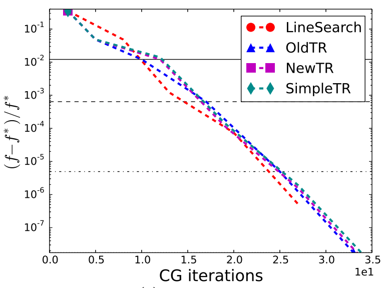
(i) `webspam`



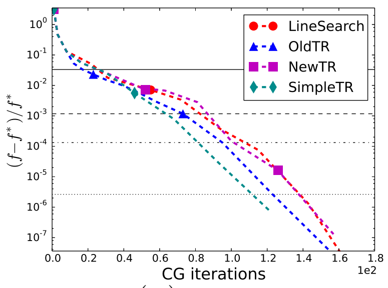
(j) `kdda`



(k) `kddb`

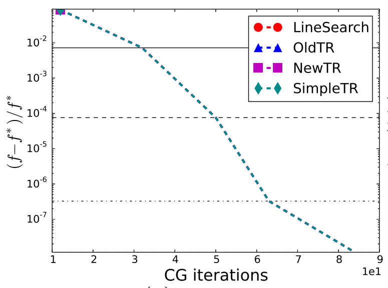


(l) `criteo`

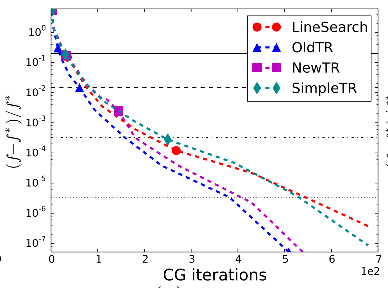


(m) `kdd12`

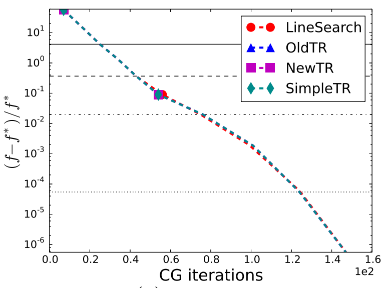
Figure 13: Comparison of line search and different trust region The x-axis shows the methods. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = 0.1C_{\text{best}}$.



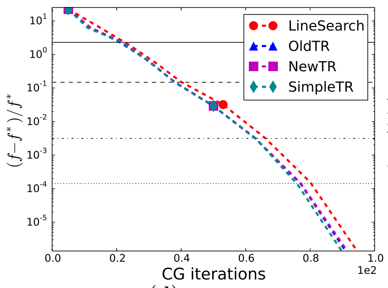
(a) HIGGS



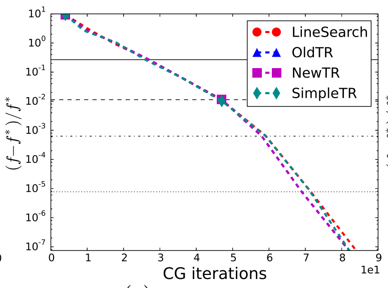
(b) w8a



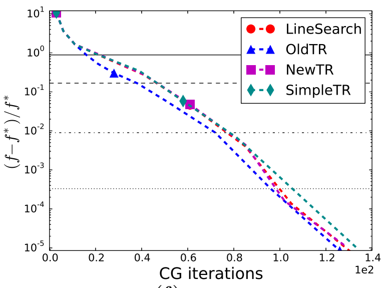
(c) news20



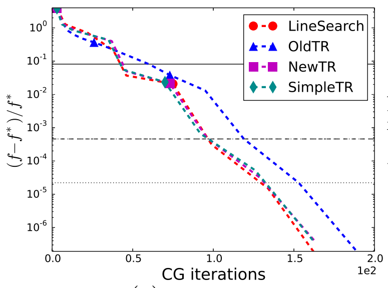
(d) rcv1



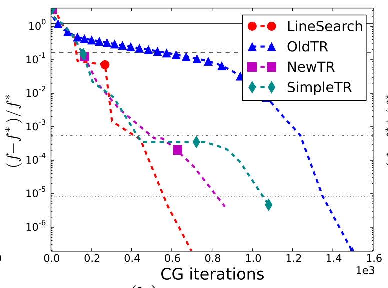
(e) real-sim



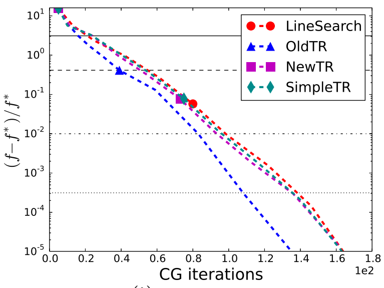
(f) url



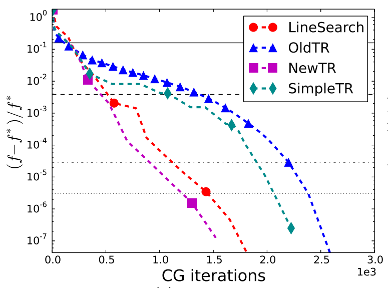
(g) yahoojp



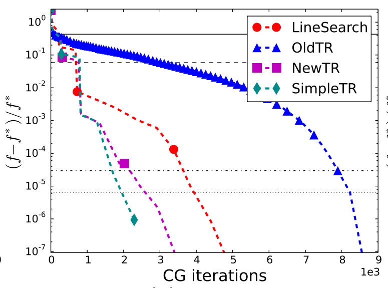
(h) yahookr



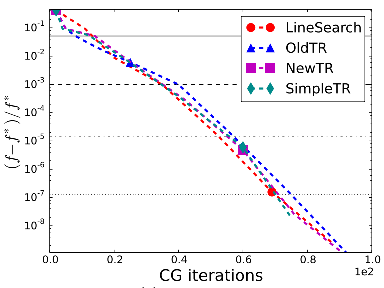
(i) webspam



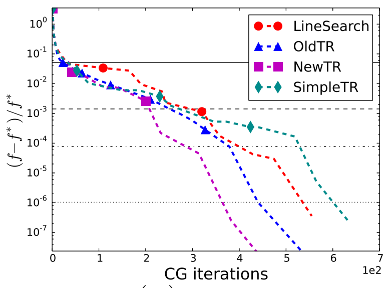
(j) kdda



(k) kddb

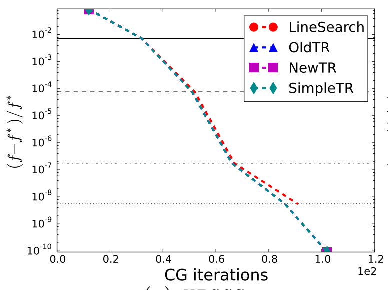


(l) criteo

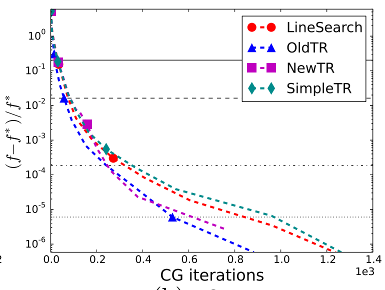


(m) kdd12

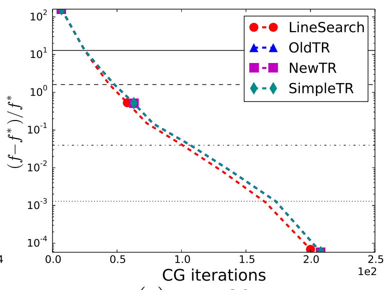
Figure 14: Comparison of line search and different trust region The x-axis shows the methods. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = C_{\text{best}}$.



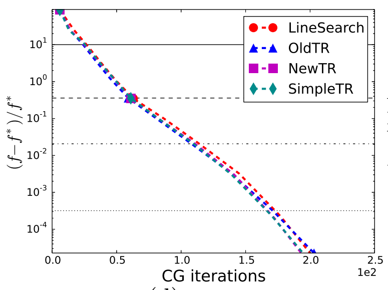
(a) `HIGGS`



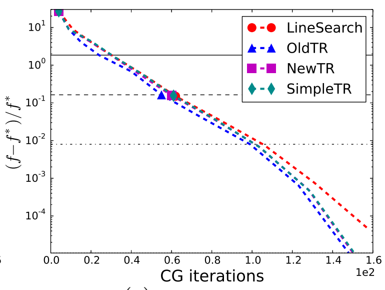
(b) `w8a`



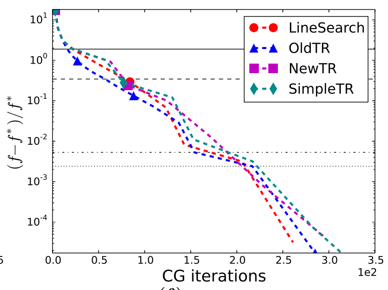
(c) `news20`



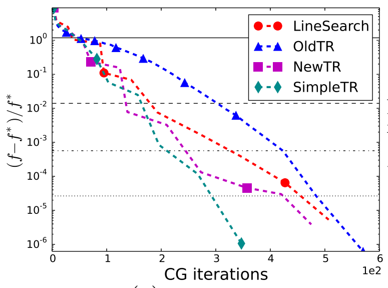
(d) `rcv1`



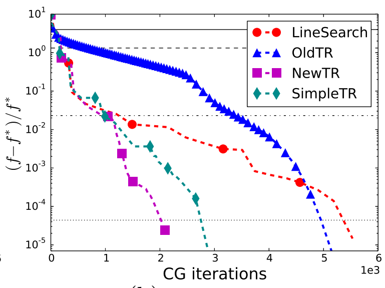
(e) `real-sim`



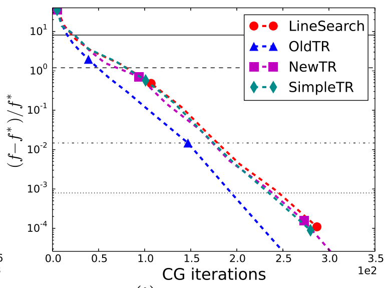
(f) `url`



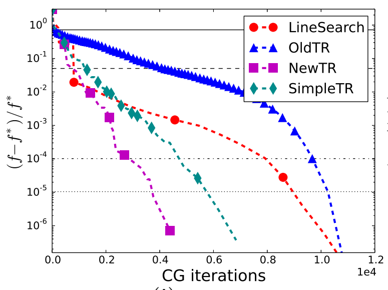
(g) `yahoojp`



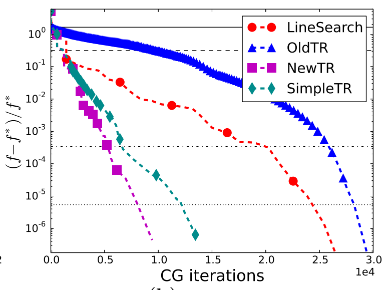
(h) `yahookr`



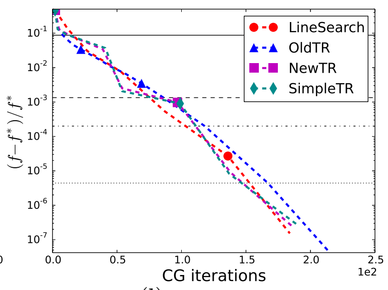
(i) `webspam`



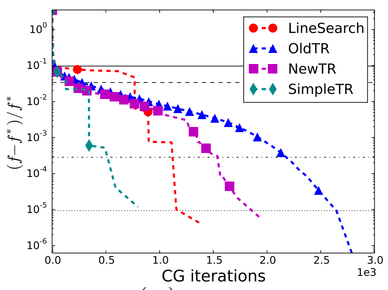
(j) `kdda`



(k) `kddb`



(l) `criteo`



(m) `kdd12`

Figure 15: Comparison of line search and different trust region The x-axis shows the methods. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = 10C_{\text{best}}$.

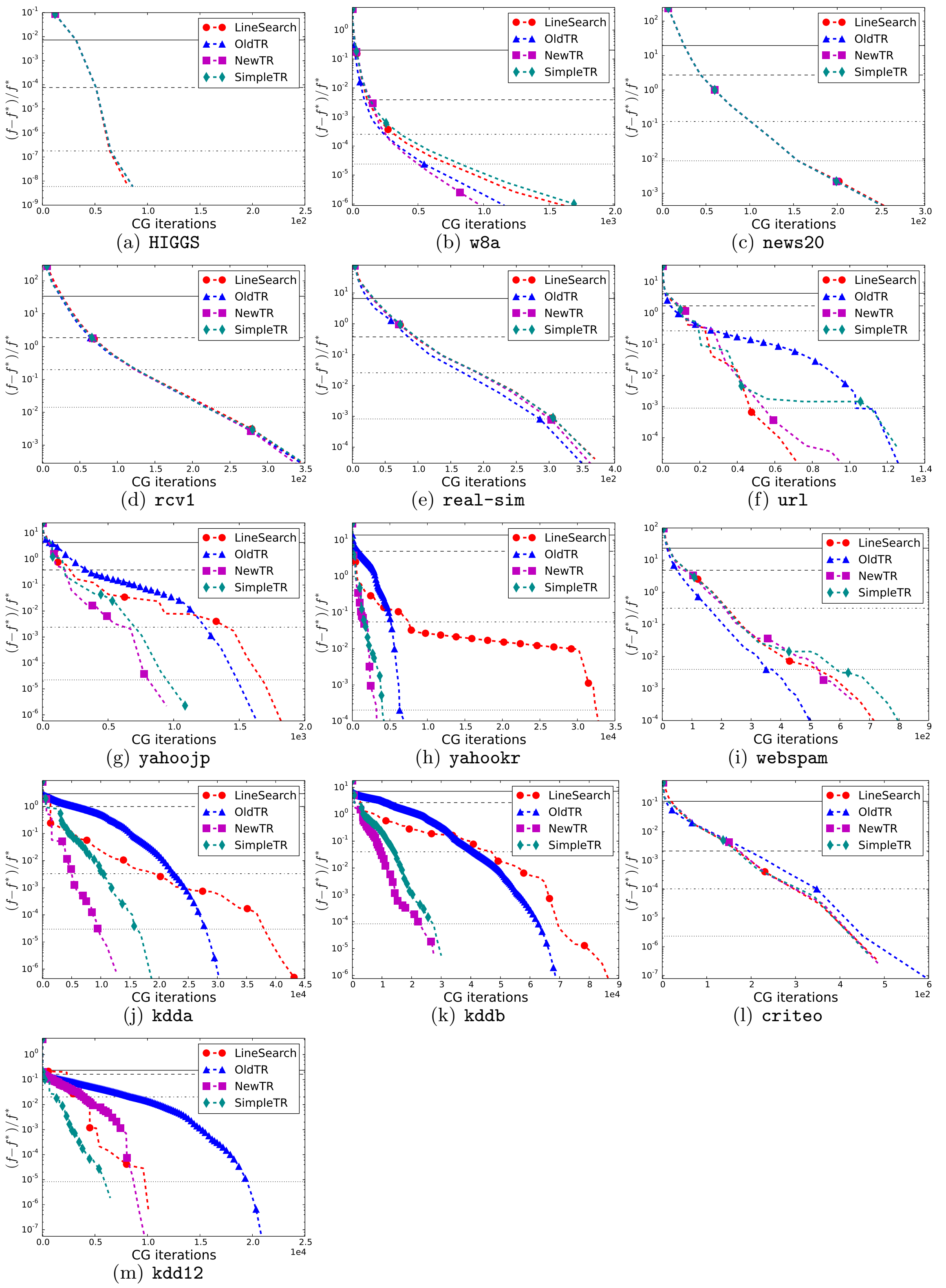
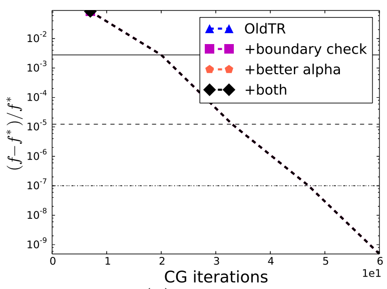
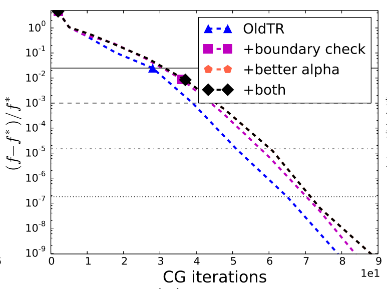


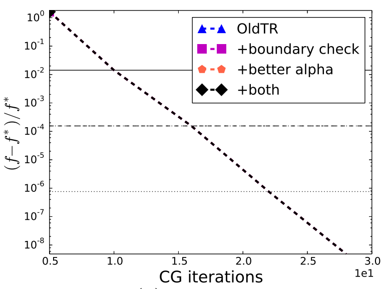
Figure 16: Comparison of line search and different trust region The x-axis shows the methods. The x-axis shows the cumulative number of CG iterations. Loss=LR and $C = 100C_{\text{best}}$.



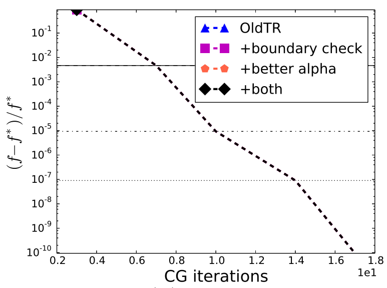
(a) `HIGGS`



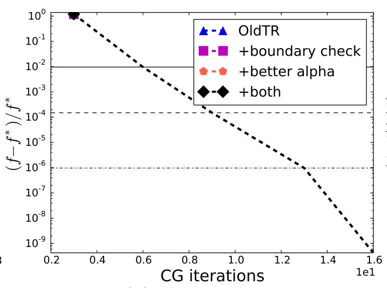
(b) `w8a`



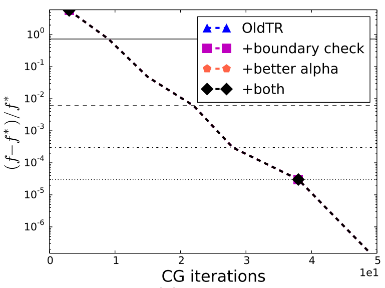
(c) `news20`



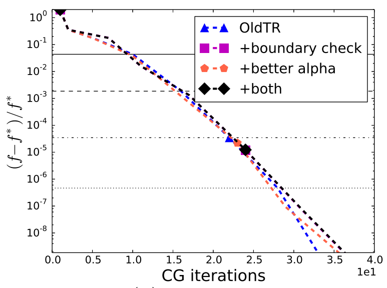
(d) `rcv1`



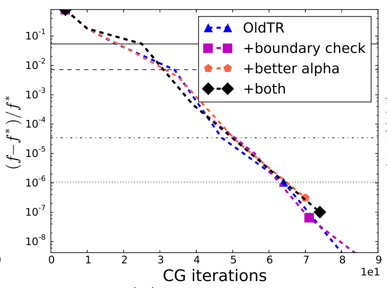
(e) `real-sim`



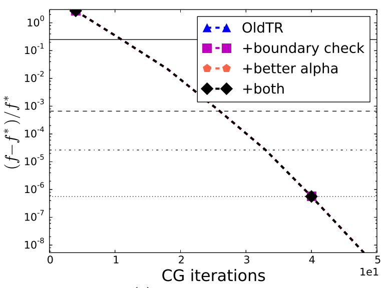
(f) `url`



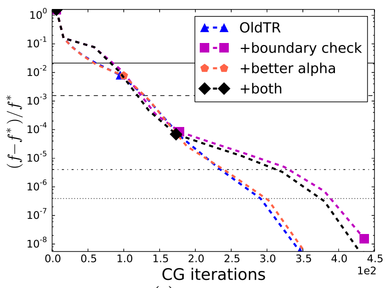
(g) `yahoojp`



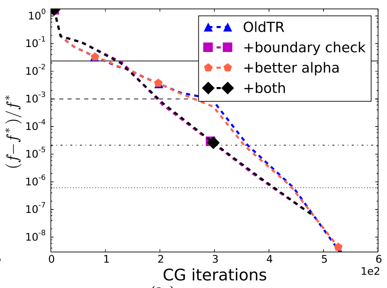
(h) `yahookr`



(i) `webspam`

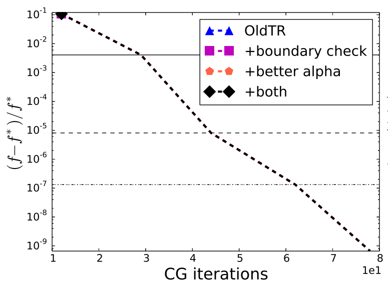


(j) `kdda`

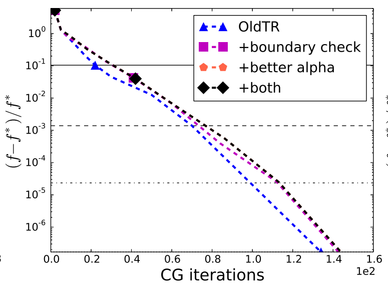


(k) `kddb`

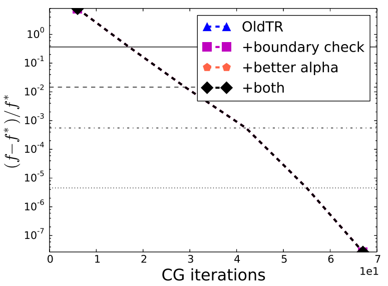
Figure 17: Convergence of using different trust region update rules. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = 0.01C_{\text{best}}$.



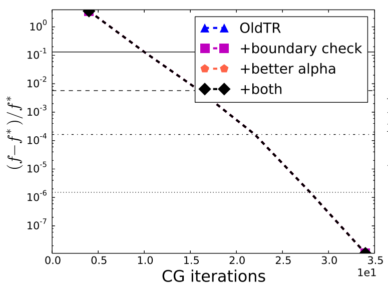
(a) `HIGGS`



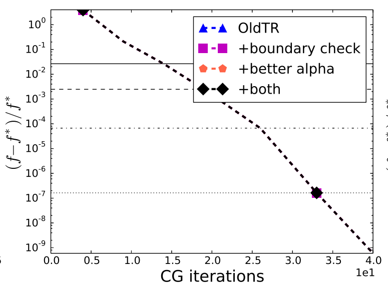
(b) `w8a`



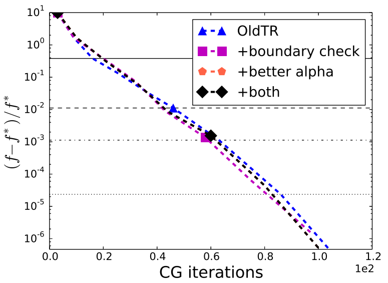
(c) `news20`



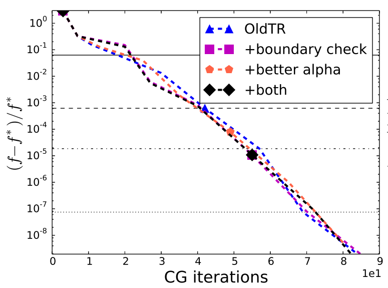
(d) `rcv1`



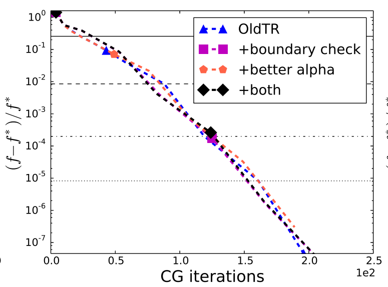
(e) `real-sim`



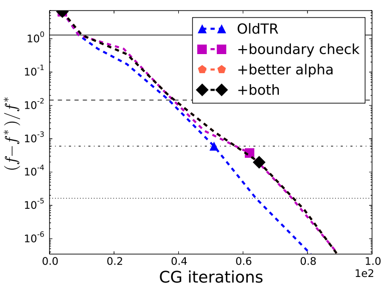
(f) `url`



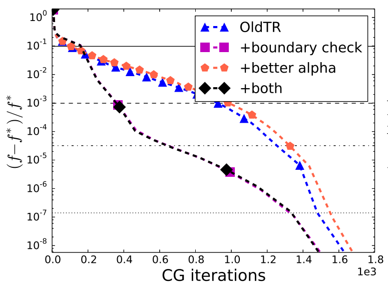
(g) `yahoojp`



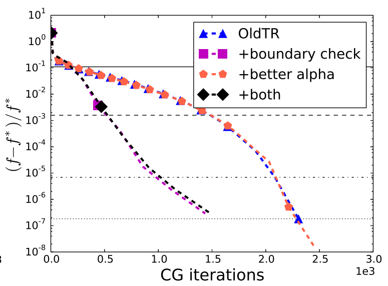
(h) `yahookr`



(i) `webspam`

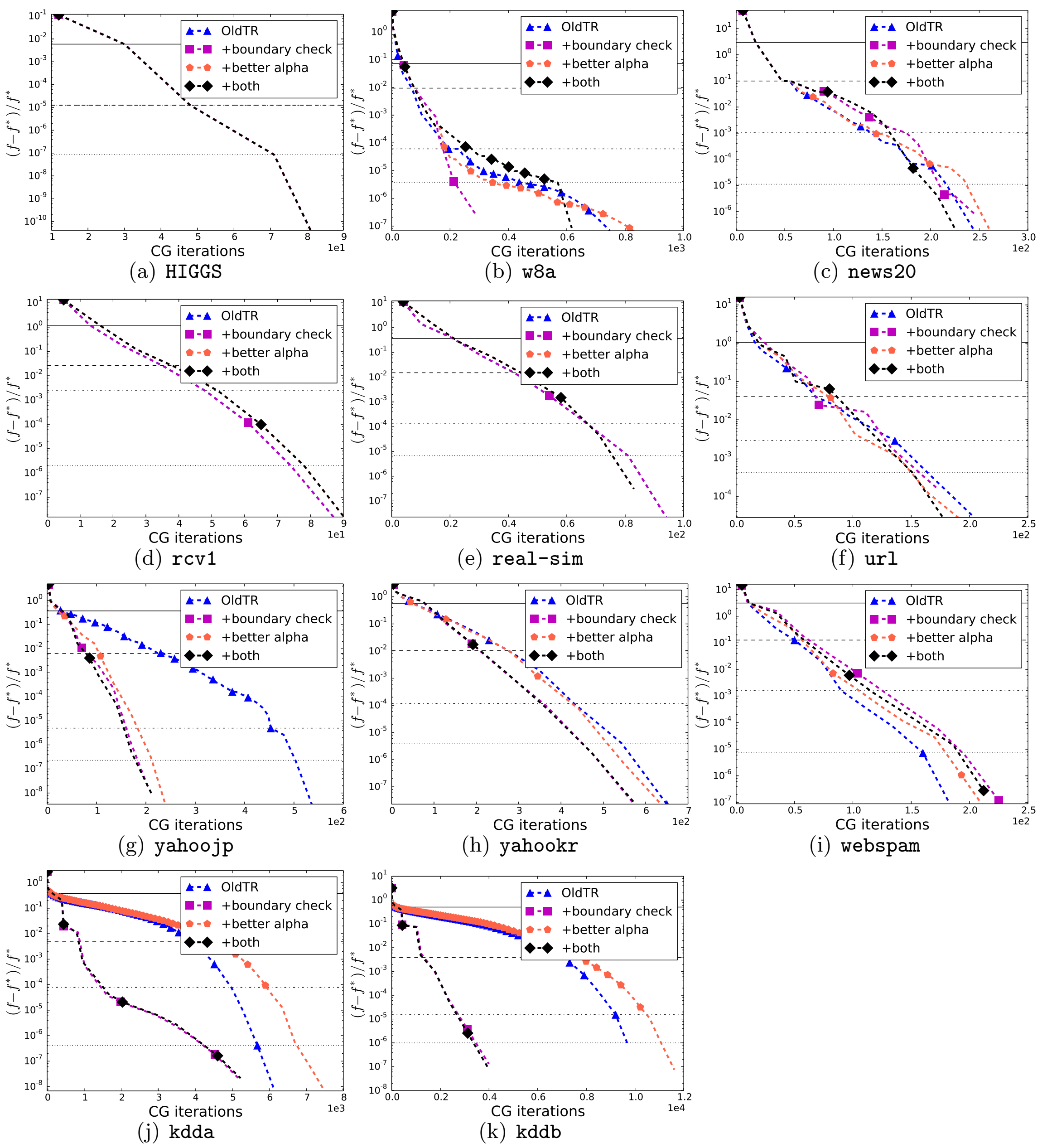


(j) `kdda`



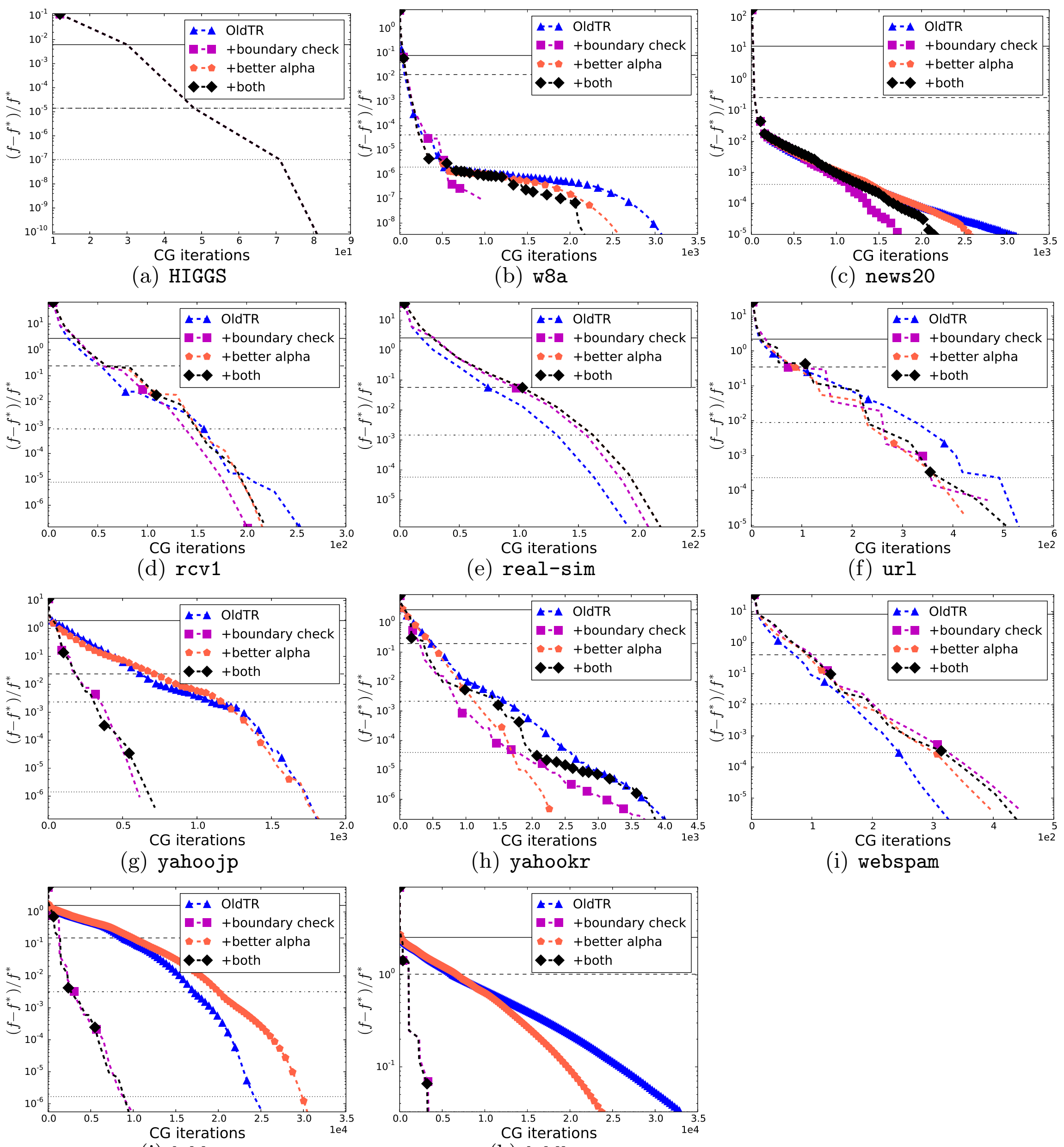
(k) `kddb`

Figure 18: Convergence of using different trust region update rules. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = 0.1C_{\text{best}}$.



(a) HIGGS (b) w8a (c) news20
(d) rcv1 (e) real-sim (f) url
(g) yahoojp (h) yahookr (i) webspam
(j) kdda (k) kddb

Figure 19: Convergence of using different trust region update rules. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = C_{\text{best}}$.



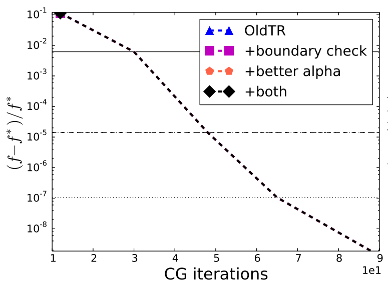
(a) `HIGGS` (b) `w8a` (c) `news20`

(d) `rcv1` (e) `real-sim` (f) `url`

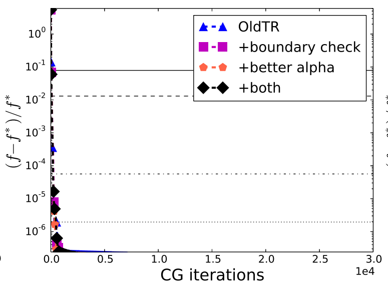
(g) `yahoojp` (h) `yahookr` (i) `webspam`

(j) `kdda` (k) `kddb`

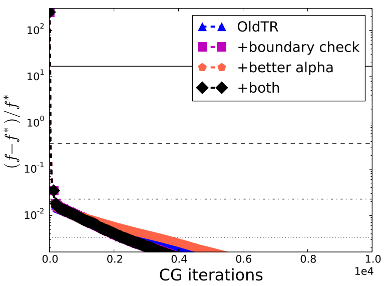
Figure 20: Convergence of using different trust region update rules. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = 10C_{\text{best}}$.



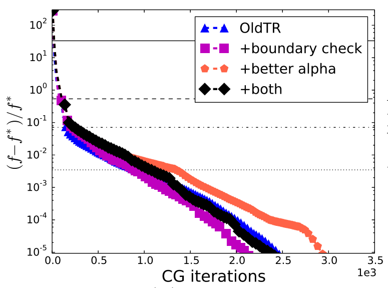
(a) HIGGS



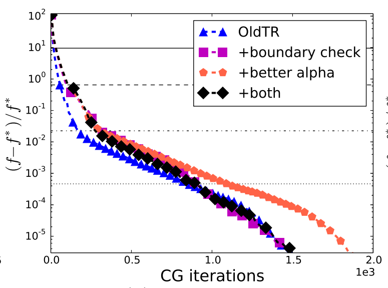
(b) w8a



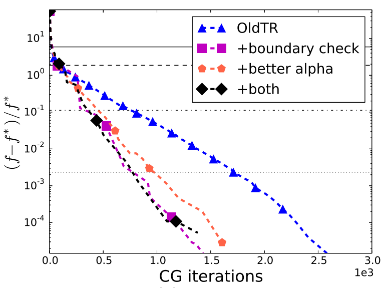
(c) news20



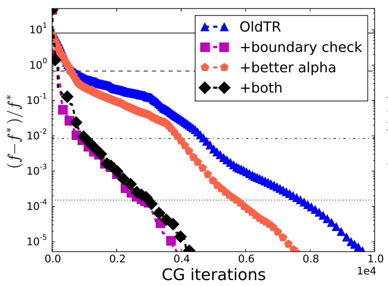
(d) rcv1



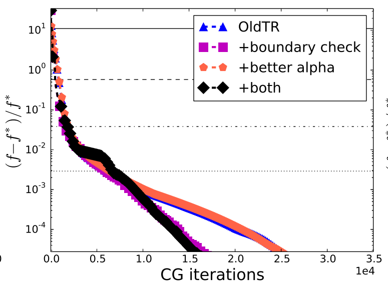
(e) real-sim



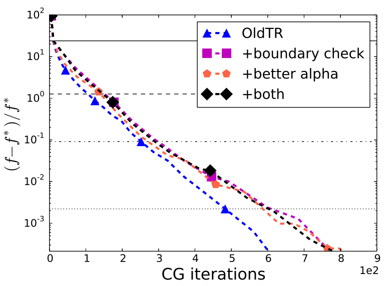
(f) url



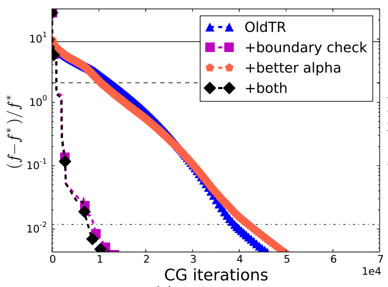
(g) yahoojp



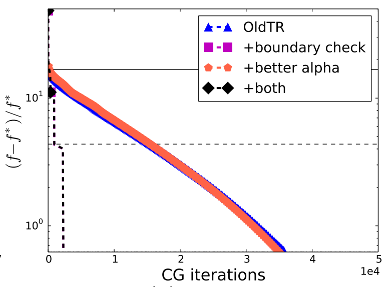
(h) yahookr



(i) webspam

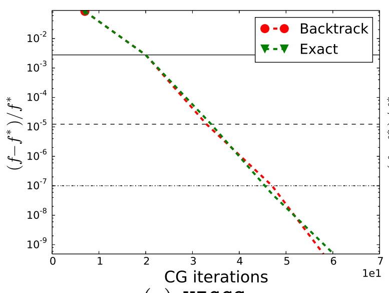


(j) kdda

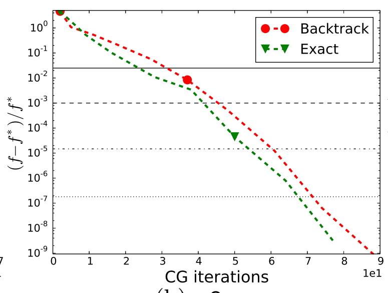


(k) kddb

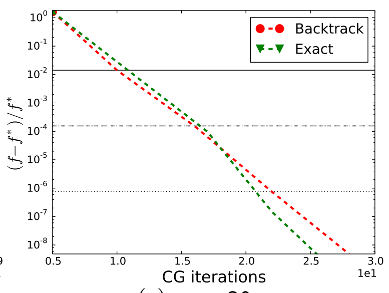
Figure 21: Convergence of using different trust region update rules. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = 100C_{\text{best}}$.



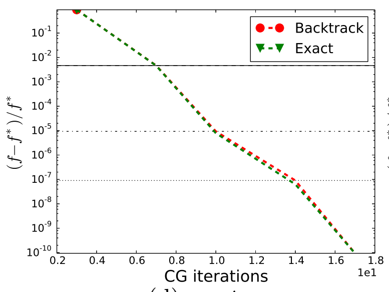
(a) `HIGGS`



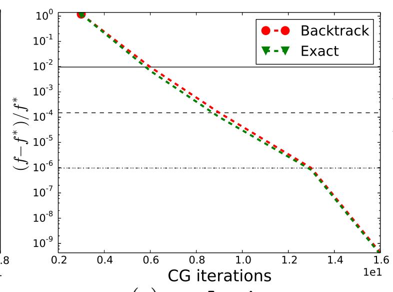
(b) `w8a`



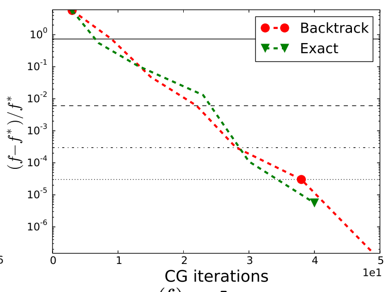
(c) `news20`



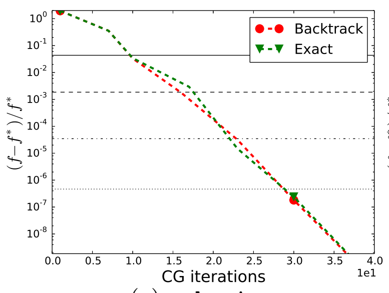
(d) `rcv1`



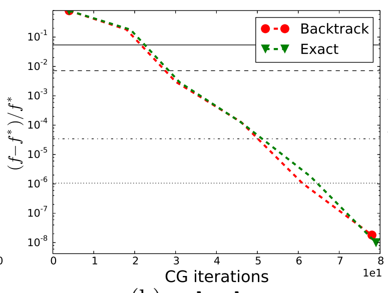
(e) `real-sim`



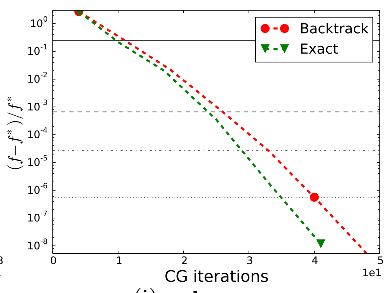
(f) `url`



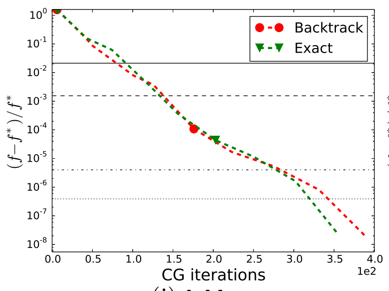
(g) `yahoojp`



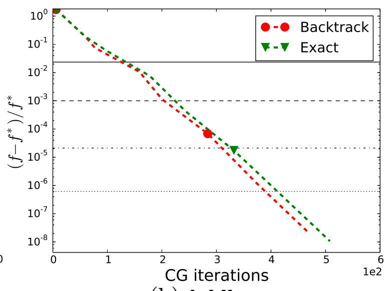
(h) `yahookr`



(i) `webspam`

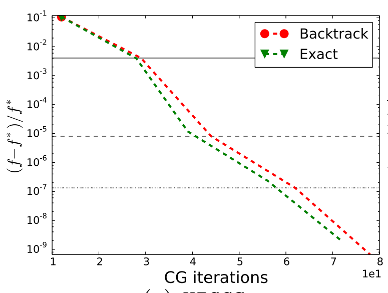


(j) `kdda`

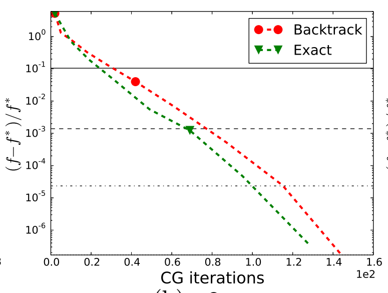


(k) `kddb`

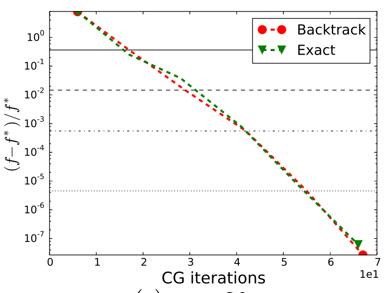
Figure 22: Comparisons of line search and exact line The x-axis shows the search. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = 0.01C_{\text{best}}$.



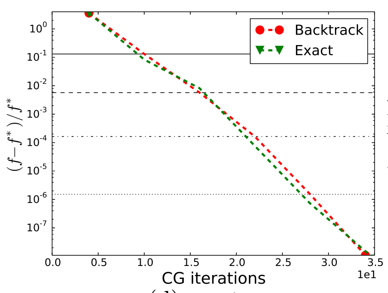
(a) `HIGGS`



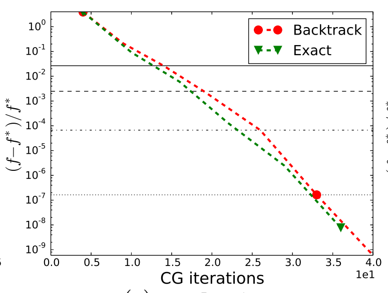
(b) `w8a`



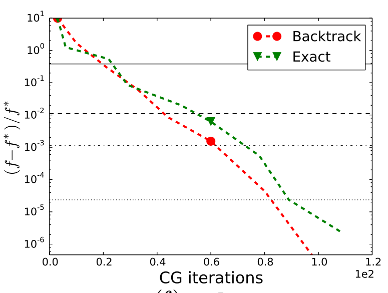
(c) `news20`



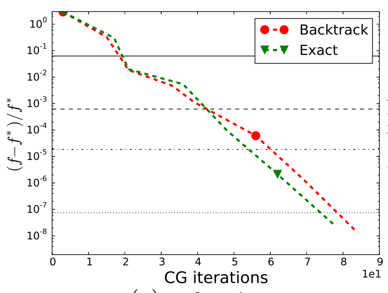
(d) `rcv1`



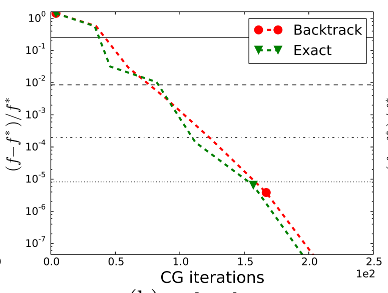
(e) `real-sim`



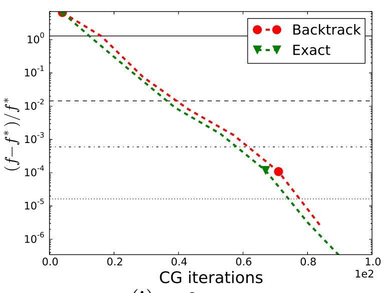
(f) `url`



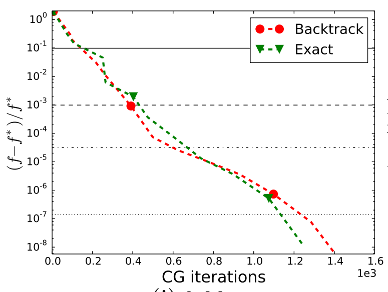
(g) `yahoojp`



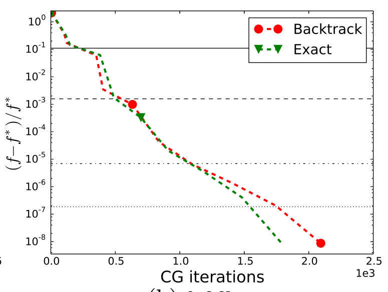
(h) `yahookr`



(i) `webspam`



(j) `kdda`



(k) `kddb`

Figure 23: Comparisons of line search and exact line The x-axis shows the search. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = 0.1C_{\text{best}}$.

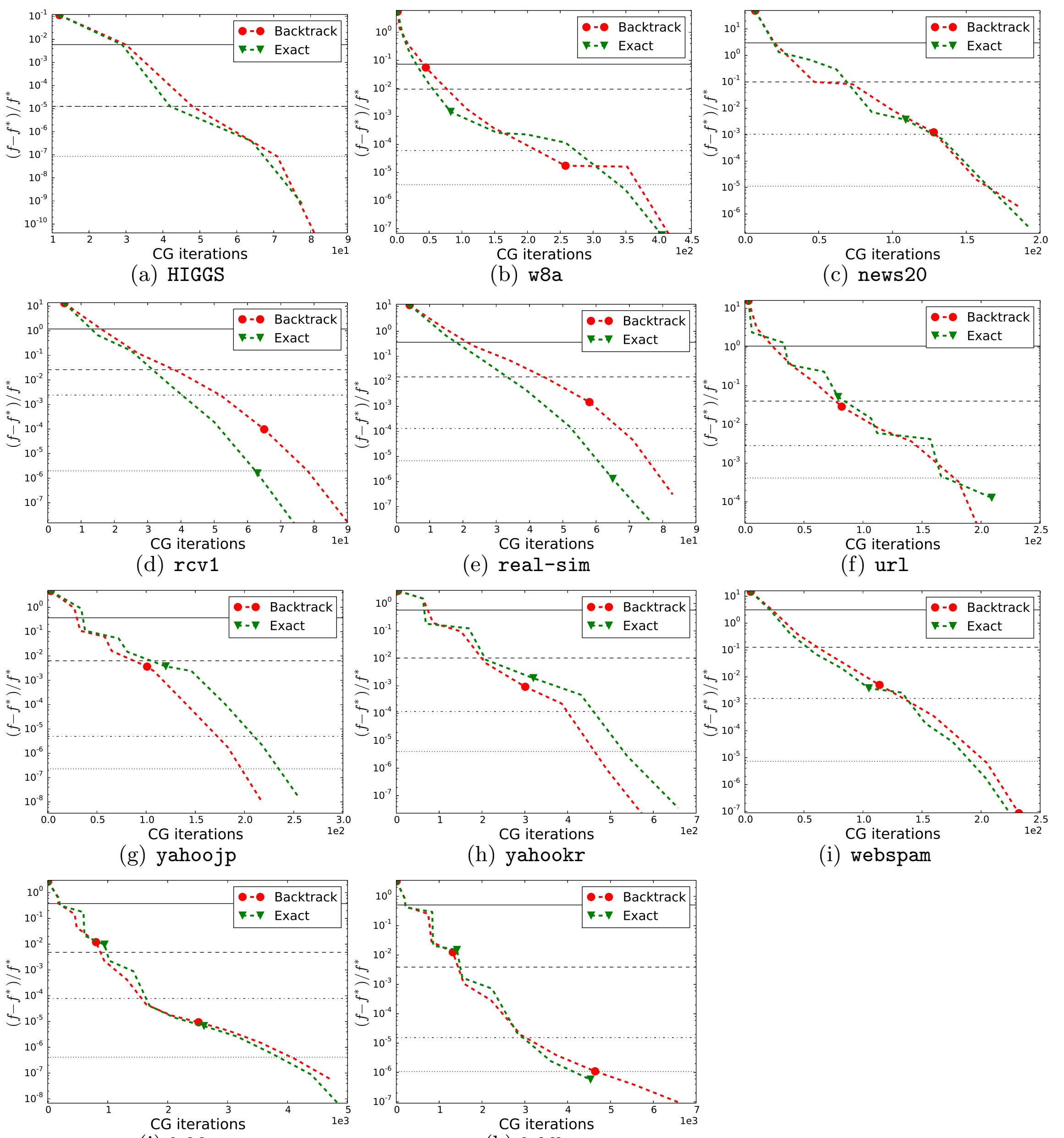
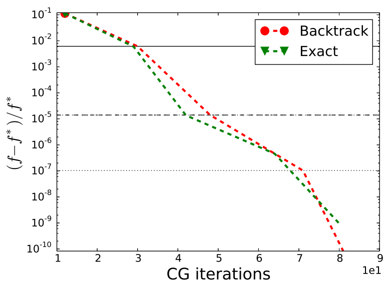
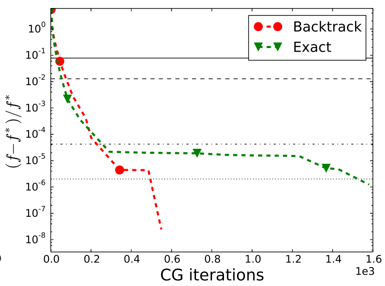


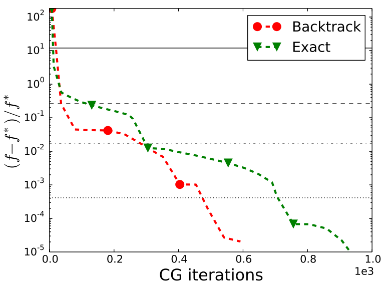
Figure 24: Comparisons of line search and exact line The x-axis shows the search. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = C_{\text{best}}$.



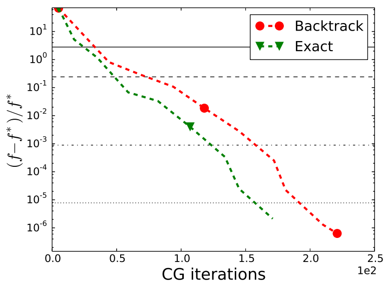
(a) `HIGGS`



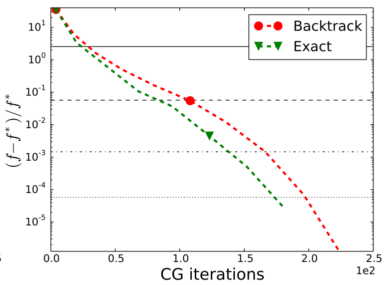
(b) `w8a`



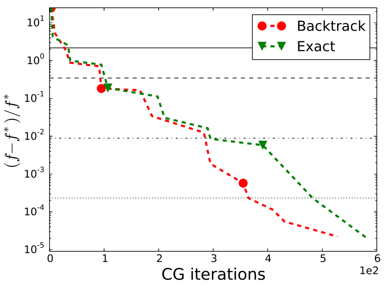
(c) `news20`



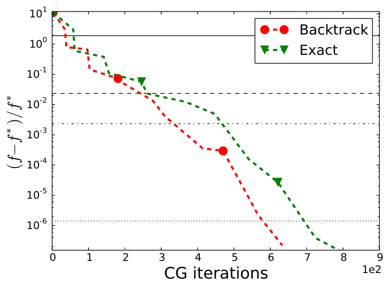
(d) `rcv1`



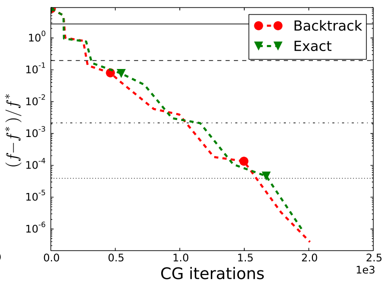
(e) `real-sim`



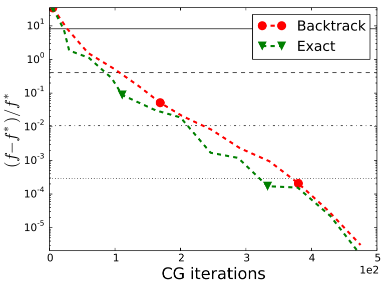
(f) `url`



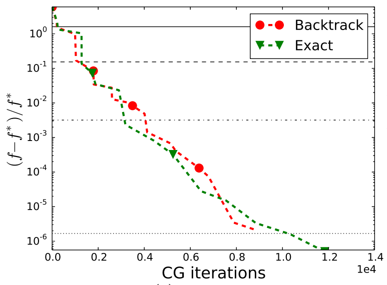
(g) `yahoojp`



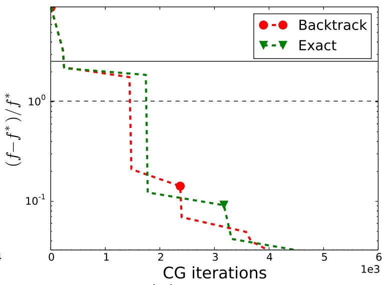
(h) `yahookr`



(i) `webspam`

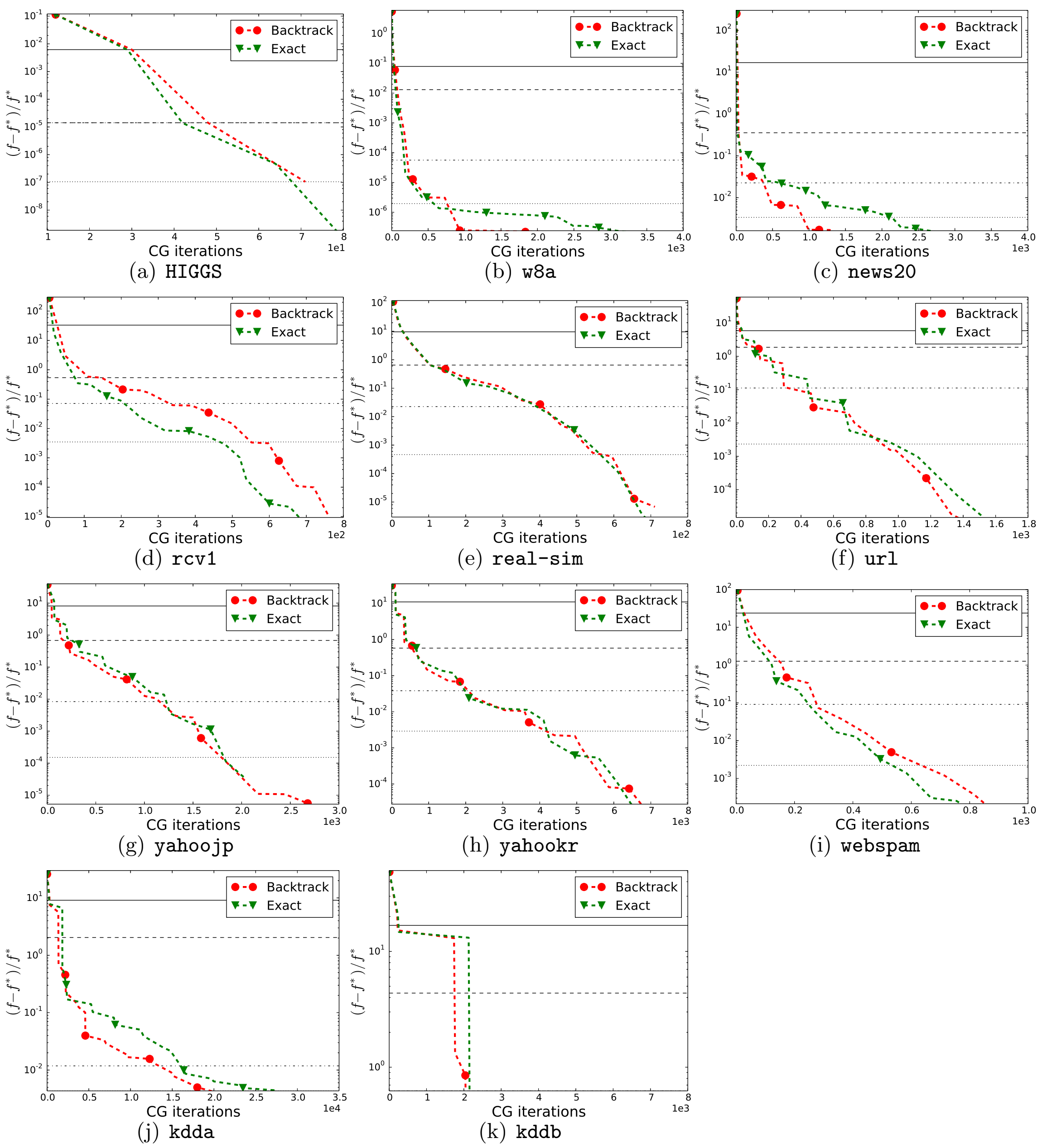


(j) `kdda`



(k) `kddb`

Figure 25: Comparisons of line search and exact line The x-axis shows the search. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = 10C_{\text{best}}$.



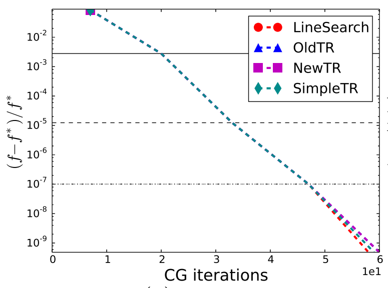
(a) `HIGGS` (b) `w8a` (c) `news20`

(d) `rcv1` (e) `real-sim` (f) `url`

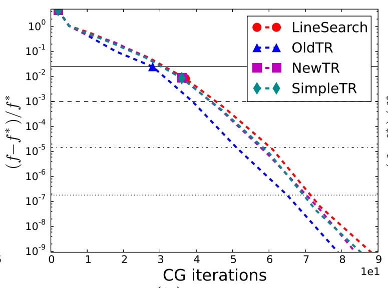
(g) `yahoojp` (h) `yahookr` (i) `webspam`

(j) `kdda` (k) `kddb`

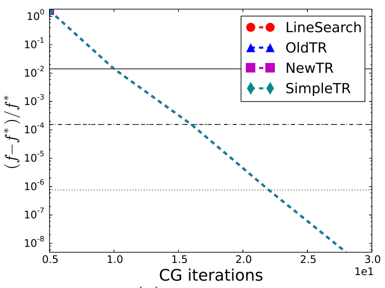
Figure 26: Comparisons of line search and exact line The x-axis shows the search. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = 100C_{\text{best}}$.



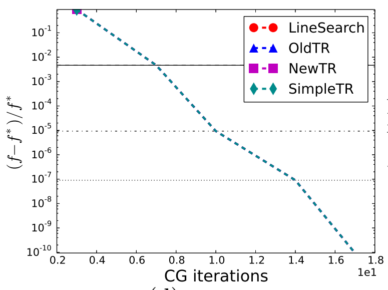
(a) HIGGS



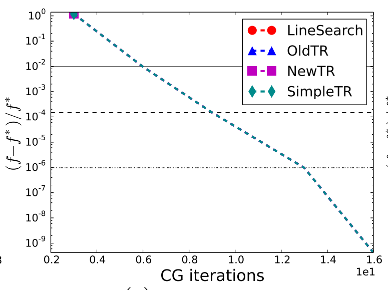
(b) w8a



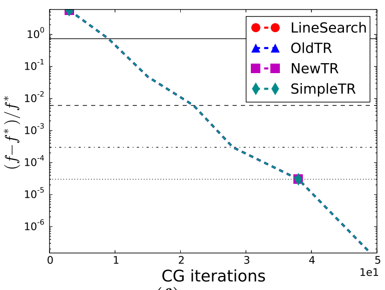
(c) news20



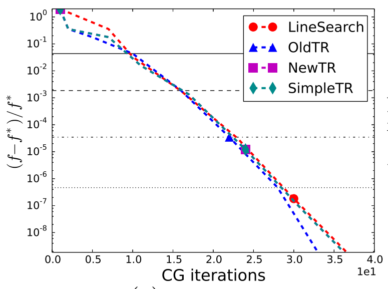
(d) rcv1



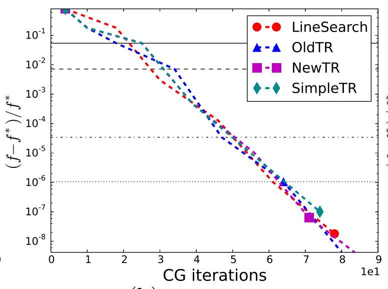
(e) real-sim



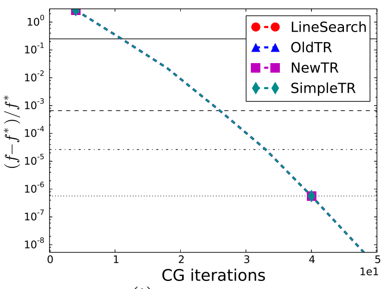
(f) url



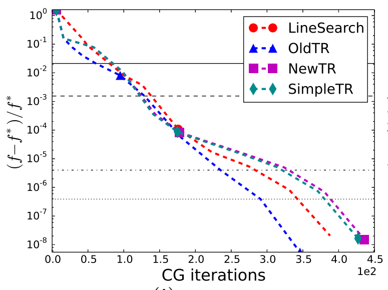
(g) yahoojp



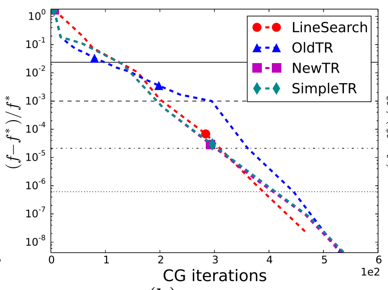
(h) yahookr



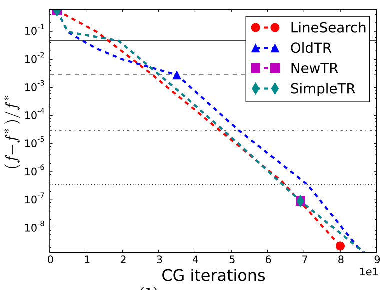
(i) webspam



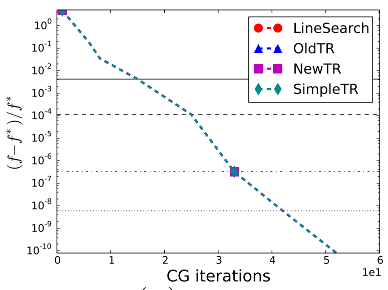
(j) kdda



(k) kddb



(l) criteo



(m) kdd12

Figure 27: Comparison of line search and different trust region The x-axis shows the methods. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = 0.01C_{\text{best}}$.

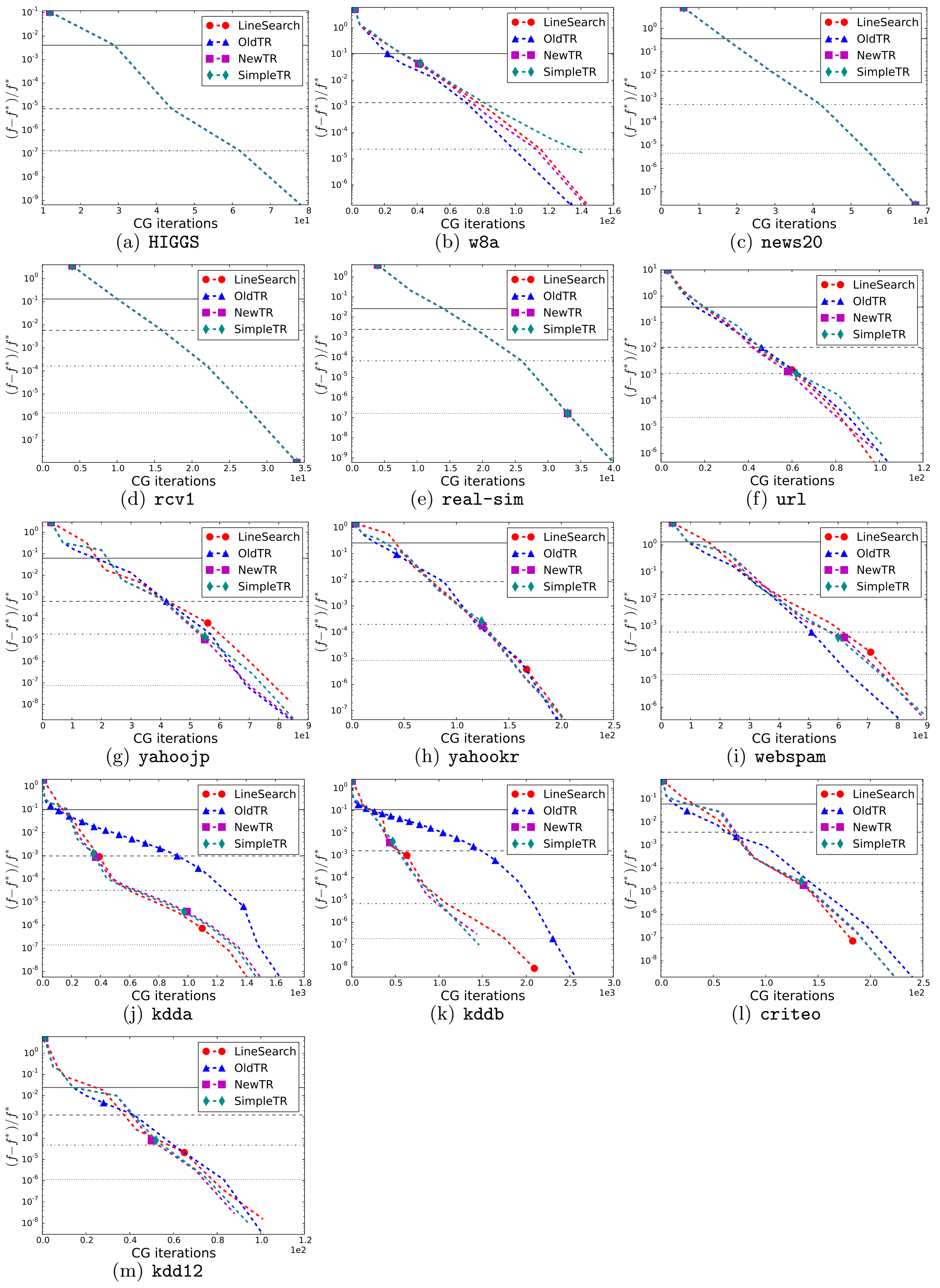
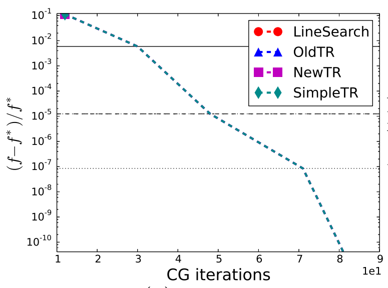
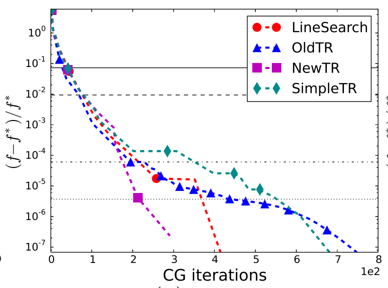


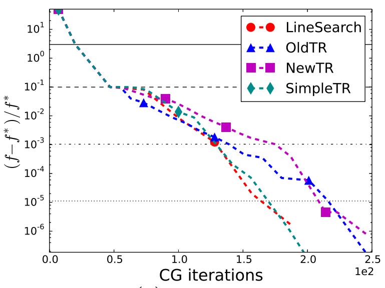
Figure 28: Comparison of line search and different trust region The x-axis shows the methods. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = 0.1C_{\text{best}}$.



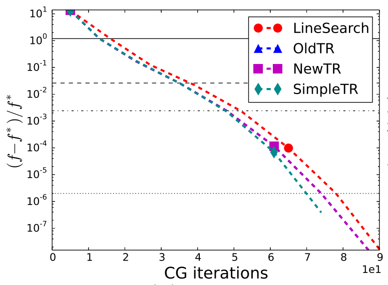
(a) `HIGGS`



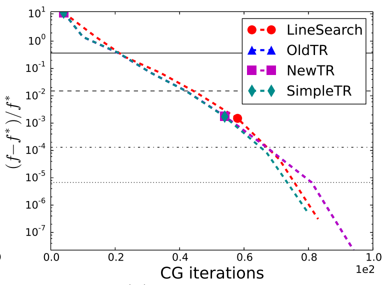
(b) `w8a`



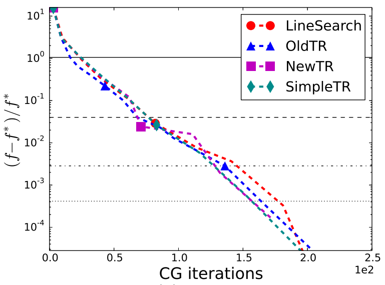
(c) `news20`



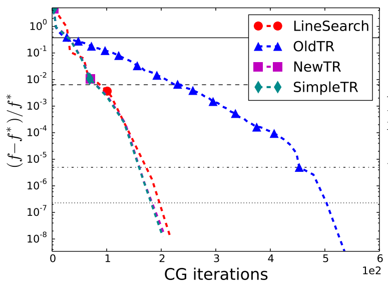
(d) `rcv1`



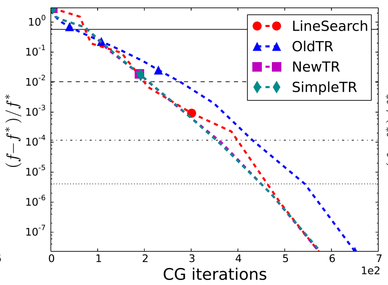
(e) `real-sim`



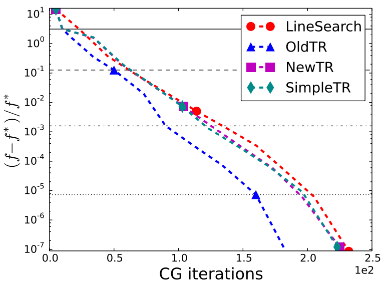
(f) `url`



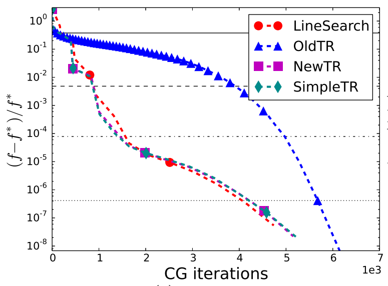
(g) `yahoojp`



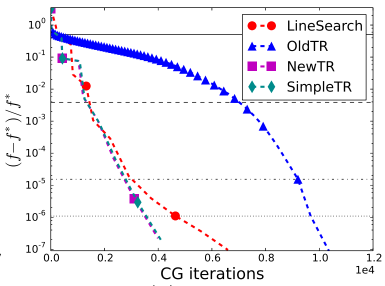
(h) `yahookr`



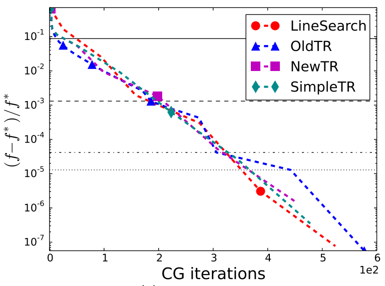
(i) `webspam`



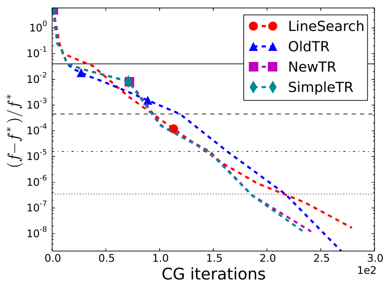
(j) `kdda`



(k) `kddb`

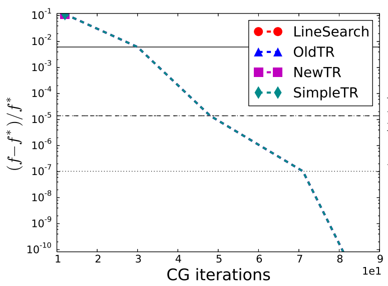


(l) `criteo`

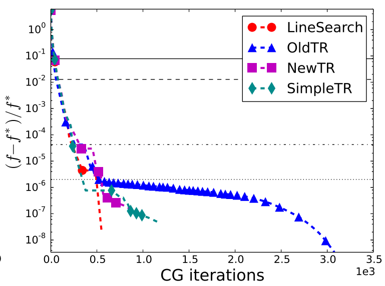


(m) `kdd12`

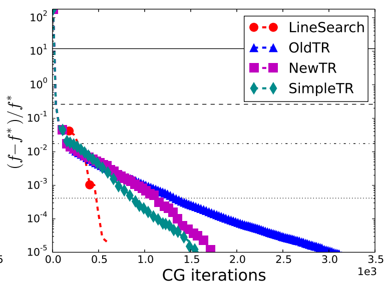
Figure 29: Comparison of line search and different trust region The x-axis shows the methods. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = C_{\text{best}}$.



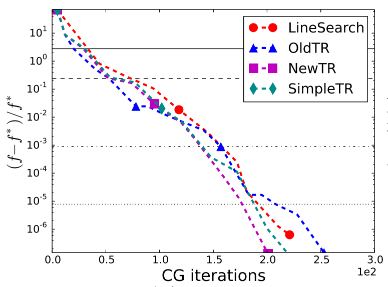
(a) HIGGS



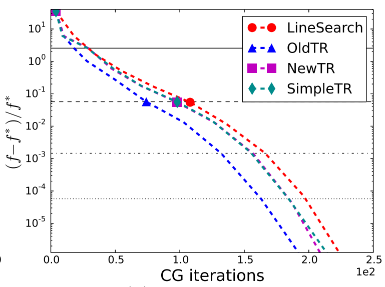
(b) w8a



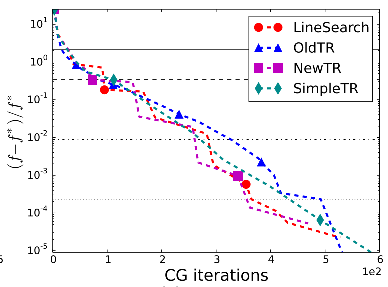
(c) news20



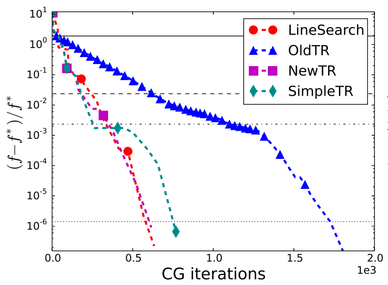
(d) rcv1



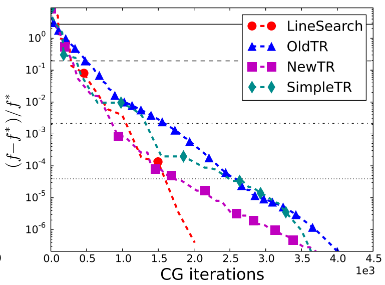
(e) real-sim



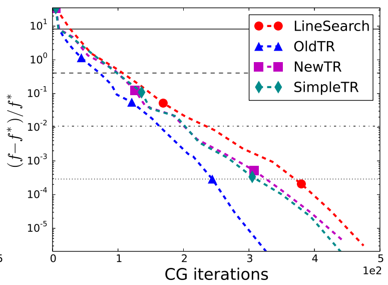
(f) url



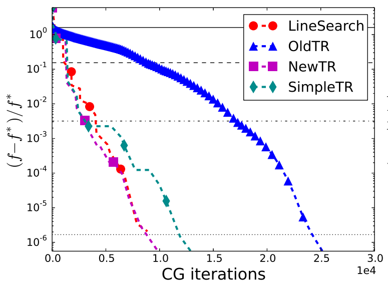
(g) yahoojp



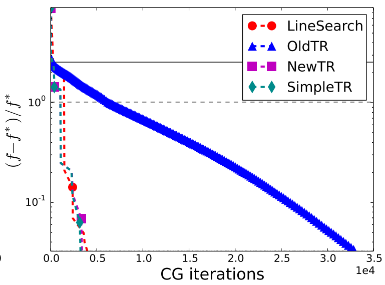
(h) yahookr



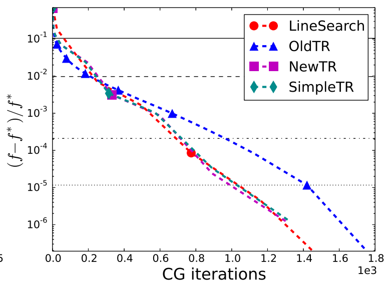
(i) webspam



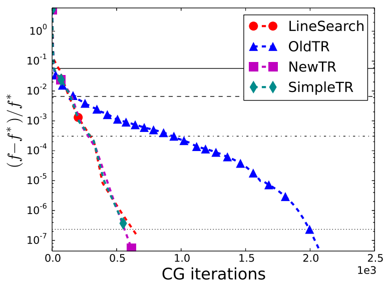
(j) kdda



(k) kddb



(l) criteo



(m) kdd12

Figure 30: Comparison of line search and different trust region The x-axis shows the methods. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = 10C_{\text{best}}$.

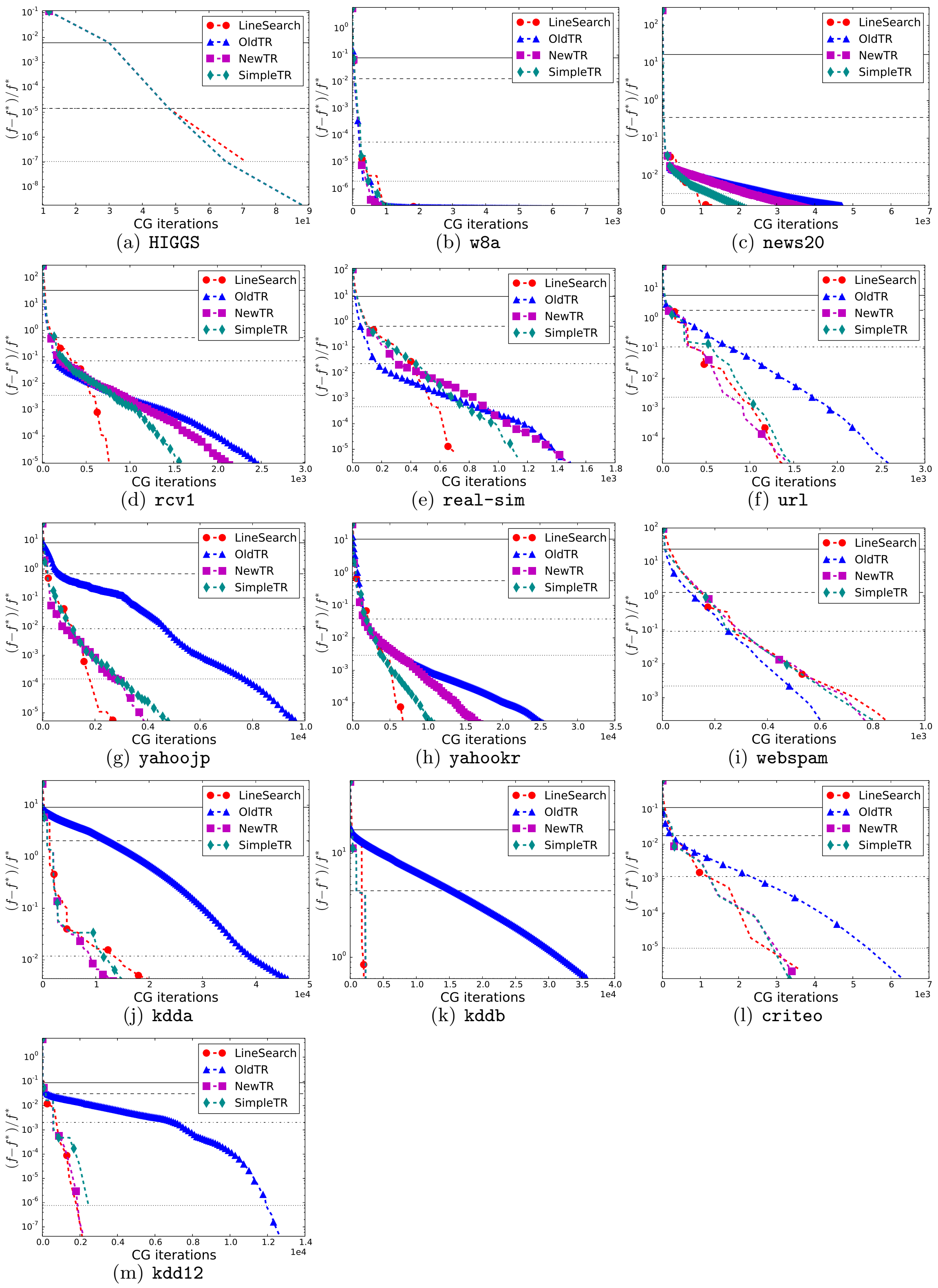


Figure 31: Comparison of line search and different trust region The x-axis shows the methods. The x-axis shows the cumulative number of CG iterations. Loss=L2 and $C = 100C_{\text{best}}$.