1. NA Simplex → NA-Simplex (e così per ogni altro NA)
2. Big-M Reformulation → scegliere se usare “formulation” “reformulation”, se usare le lettere maiuscole o minuscole. C’è confusione in merito (lo stesso per Convex-Hull)
3. I Big-M → I-Big-M (a volte manca anche il secondo trattino)
4. archimedean → Archimedean (in caso fosse necessario: non-Archimedean)
5. Non usare abbreviazioni (es.: didn’t → did not)
6. Ad inizio sezione 1.1 avete scritto che l’ottimo è -20 e solo alla fine spiegate perché il risultato vi è venuto 20. Cambiate il tipo di ottimizzazione (da min a max o viceversa), cambiate il segno della funzione obiettivo e tutto viene più lineare (modificare il resto del paper di conseguenza)
7. pag 5 line 1 algorithm → algorithms
8. Ad inizio sezione 1.3 avete formulato il problema con una “or” nella funzione obiettivo. Sebbene capisca cosa intendiate non è formale. Scrivete I due problemi separatamente (usate l’environment minipage così stanno uno accanto all’altro)
9. pg 10 line 6 variable → variables
10. Poco prima di 1.6.2 Discutere del perché del diverso numero di iterazioni richieste (è abbastanza ovvio ma un lettore non attento [a buon intenditore poche parole] può rimanerne confuso). Può essere giusto spendere una riga anche sul perché di una sola iterazione nel caso di ottimizzazione di (-1, 1). Questo non è affatto ovvio per coloro che non conosco l’I-Big-M.
11. Pg 11 riformulazione Convex-Hull: I vincoli x_ij >= 0 sono ridondanti per il solver che state utilizzando. Se li volete tenere, metteteli alla fine dei vincoli e decidete se mettere una riga dove specifica la loro ridondanza. N.B. I vincoli di >=0 delle variabili o si mettono tutti o non se ne mette nessuno, difatti mancano quelli di x_i e di y. Se non li mettete specificate invece che si assumono tutte le variabili positive (se altrove si ripete questa incongruenza aggiustatela)
12. Pag 16. Aggiungete la versione KKT e matriciale del problema di programmazione semidefinita nel caso specifico delle due regioni disgiunte che state studiando
13. triditional Simplex → Qualora vi riferiate a quello standard utilizzate “standard”, se vi riferite all’NA-Simplex utilizzate “NA-Simplex” e quello nuovo sarà “NA-Simplex2”
14. modify è un verbo, il sostantivo è modification
15. Sottosezione 5.2.1→ troppo corta, accorpatela a quella precedente
16. pg 26 sezione 6.1→ la funzione può essere riscritta (fatelo) in maniera più semplice e concisa usando le lambda functions: return map(x→round(std(x), digits=5), BAN_vector). Ho abbozzato, potrebbe essere imprecisa ma l’idea è questa
17. pag 27 sezione 6.3 Scrivere due righe dove si commenta il fatto che, anziché settare alpha^2 hard-coded, sarebbe più corretto implementare una routine che scelga la potenza di alpha usando uno dei lemmi (vedete voi quale) che porta alla generalizzazione del teorema di Megiddo per casi non-standard (ne abbiamo parlato in merito all’NA-IPM ma in realtà è un risultato che vale in generale per tutti i sistemi lineari)
18. Costruire un benchamrk non banale (ovvero dove le regioni disjunte hanno più di 4 vertici e nessun lato parallelo agli assi) con più di 2 regioni disgiunte (almeno 4 direi). [So che questo è parzialemente già coperto dai benchmark reali ma: 1) il ragionamento è più lineare e coerente perché segue un approccio scientifico (prima si testano più regioni su un problema artificiale e poi si passa al problema reale); 2) un lettore disattento potrebbe non accorgersi che questo caso è trattato nei problemi reali e comunque vorrebbe più di un benchmark di prova così da essere certo che non sia un caso fortuito che tutto sia tornato]
19. Nei problemi reali specificate bene il numero di regioni disgiunte coinvolte nel problema, commentando il perché sono tali (es.: da dove vengono fuori, che significato hanno, ecc)