Om onderzoek te doen naar de gevoeligheid van ILS voor de waarde van S moesten we eerst weten welke waardes voor S relevant waren. Dit hebben we gedaan door de eerste en makkelijkste 9x9 sudoku 100 keer op te laten lossen op elke waarde van S tot 30. Als de waarde van S hoger werd dan 20, begon het aantal iteraties en de looptijd ernstig toe te nemen. Ook het aantal sudoku’s dat niet meer opgelost konden worden steeg na S20. Na dit eerste experiment hebben we het ILS algoritme de andere sudoku’s 50 keer laten oplossen voor alle S waarden tot en met 20. De tijd die het koste om grote sudoku’s op te lossen bleek echter te lang te zijn om 1000 keer te doen, dus hebben we ervoor gekozen om de grote sudoku’s (3,4,14,15) buiten beschouwing te laten bij het onderzoek naar de gevoeligheid van de S waarde. Nadat het programma klaar was vielen er een paar dingen op.

Bij zowel het aantal iteraties als de tijd die nodig is om te testen is de waarde hoger bij S1 dan bij de volgende S waardes. Dit is te verklaren omdat een enkele switch waarschijnlijk niet genoeg is om veel verandering te veroorzaken. Wat ook opviel, en wat mogelijk een probleem is voor het onderzoek is dat het aantal sudoku’s dat het maximum aantal random walks overschreed voordat er een oplossing was gevonden groter bleek dan gedacht. Dit maximum was gekoppeld aan de initiele score van de sudoku, dus hoe moeilijker deze was, hoe vaker de sudoku een random walk kon doen, maar vooral bij moeilijke sudoku’s was er een hoog percentage sudoku’s dat on opgelost bleef. Wanneer deze resultaten van alle sudoku’s gestapeld werden ontstond er een interessante grafiek. Hier is te zien dat bij een s waarde 1, er veel sudoku’s onopgelost blijven, en dat er een dal is tussen S3 S8 waarna het weer stijgt. Het hoge percentage van sudoku’s dat on opgelost blijft is een probleem, omdat dit zorgt voor een nivellerend effect bij het aantal iteraties en de looptijd.

Om dit probleem op te lossen hebben we het maximum aantal random walks aangepast zodat dit bij alle sudoku’s van een bepaalde grootte een zelfde waarde heeft, dat groter is dan de mogelijke moeilijkheidsgraad. Na deze aanpassing hebben we alle 9x9 sudoku’s opnieuw 20 keer uitgevoerd op alle S waarden tussen de 1 en 15, omdat het uit eerdere resultaten duidelijk leek dat een S hoger dan 15 de looptijd en het aantal iteraties alleen maar hoger zou maken. De aanpassingen aan het algoritme hebben het een factor 2 langzamer gemaakt doordat het nu in het slechtste geval soms meer dan 2 keer langer door laat gaan, maar het aantal onopgeloste sudoku’s is ongeveer gehalveerd. Voor dit onderzoek is het niet erg dat het algoritme langer doorloopt, omdat dit meer informatie geeft dan als het voortijdig stopt.

Wat bij zowel de resultaten uit het eerste onderzoek als die van het tweede te zien is als je de resultaten opstapelt, is een golf beweging die oploopt. De ligging van de toppen en dalen van de golven zijn bij de resultaten van het tweede onderzoek beter te zien, waarschijnlijk doordat de resultaten minder genivelleerd zijn, maar ook bij het eerste onderzoek zijn ze op ruwweg dezelfde plaatsen te vinden rond de 5, de 8 en de 11. Bij het kiezen van een S waarde moet er rekening gehouden worden met twee dingen. Als eerste wil je een S waarde die niet te laag is. Zoals eerder gezegd, is er minder kans dat er een significante wissel plaats vind als er minder gewisseld wordt. Ten tweede is het belangrijk dat de s waarde niet te hoog is. Elke wissel brengt kosten met zich mee en als er te vaak gewisseld wordt, is er een grotere kans dat de wissels ervoor zorgen dat er werk opnieuw gedaan moet worden. Uit de resultaten die bij dit onderzoek verkregen zijn valt te concluderen dat een S waarde van 5 waarschijnlijk een goede keuze is voor optimale resultaten.

Sudoku 1 average runtime 105.999946594 ms, back tracking 0.999927 ms

Sudoku 2 average runtime 101591.66669 ms, backtracking 1514.99986649 ms

Sudoku 3 average runtime 83970.5001711 ms, backtracking > 300000 ms

Sudoku 4 average runtime 17496.5555138 ms, backtracking > 300000 ms

Sudoku 5 average runtime 17598.3539579 ms, backtracking 1060.9998703 ms

Sudoku 6 average runtime 3953.91175326 ms, backtracking 489.000082016 ms

Sudoku 7 average runtime 16367.4545721 ms, backtracking 1269.00005341 ms

Sudoku 8 average runtime 13761.6427966 ms, backtracking 104.000091553 ms

Sudoku 9 average runtime 1557.59998427 ms, backtracking 24.9998569489 ms

Sudoku 10 average runtime 1037.2916807 ms, backtracking 125.999927521 ms

Sudoku 11 average runtime 12684.799989 ms, backtracking 383.999824524 ms

Sudoku 12 average runtime 2854.7179515 ms, backtracking 42.0000553131 ms

Sudoku 13 average runtime 442573.99964 ms, backtracking > 600000 ms

Sudoku 14 average runtime 305625.00047 ms, backtracking 13164.9999619 ms

Sudoku 15 average runtime, 300137.9976 ms, backtracking 104321.0000999 ms

De gemiddelde runtime van ILS is over het algemeen 10 tot 100 keer groter dan die van het reverse-backtracking algoritme. Dit is best logisch, omdat ILS niet volledig is en er dus een kans is dat de oplossing van de sudoku niet in een run gevonden wordt. Wanneer de oplossing niet gevonden wordt moet het programma gestopt worden door een andere stopcriteria. Bij dit practicum is de extra stop criterium dat een maximum aantal keer een random walk is gedaan. Dit maximum bepalen was lastig, want bij een te laag maximum werd het globale optimum vaak niet gevonden, maar als het maximum te groot is, zorgt dit ervoor dat de runtime onhandelbaar wordt. Hier is gekozen voor een dynamisch maximum gebaseerd op de initiële score van de sudoku. De pogingen die niet uitkomen op een oplossing tellen wel mee bij de looptijd, waardoor de reëele tijd die het kost een stuk lager kan uitpakken als de oplossing snel gevonden wordt. ILS kost polynomiaal veel tijd waar backtracking exponentieel veel tijd kost. Voor weinig variabelen is backtracking voordeliger, maar zodra de sudoku’s groter worden, is het niet meer mogelijk om ze met backtracking op te lossen, omdat het teveel tijd kost. ILS is geschikter om grotere sudoku’s op te lossen, maar bij sudoku’s waarbij er veel lokale optima dicht rond het globale optimum liggen komt ook ILS in de problemen. Dit is het geval bij sudoku’s 13, 14 en 15. Het zijn kleine sudoku’s waarbij 64 van de 81 vakjes leeg zijn, het maximale aantal waarbij ze nog oplosbaar zijn. ILS vindt hierbij maar bij 1 op de 100 pogingen een oplossing, waardoor de tijd die nodig is erg oploopt. Bij sudoku 4 zijn 208 van de 256 vakjes leeg, maar hier wordt bij bijna elke poging een oplossing gevonden. Wat het verschil precies maakt is moeilijk te zeggen, maar het is wel een opvallend gegeven.

\ref{appendix:sudoku1} Sudoku 1 & Forward Checking & Snelste backtracking(RB) \\

Recursions & 41 & 57 \\

Run Time & 13.0000114441 ms & 0.999927520752 ms \\

\end{tabular}

\\

\begin{tabular}{lll}

\ref{appendix:sudoku2} Sudoku 2 & Forward Checking & Snelste backtracking(RB) \\

Recursions & 363 & 42345 \\

Run Time & 796.000003815 ms & 1514.99986649 ms \\

\end{tabular}

\\

\begin{tabular}{lll}

\ref{appendix:sudoku3} Sudoku 3 & Forward Checking & Snelste backtracking(-) \\

Recursions & ? & ? \\

Run Time & $>$ 600000.000 ms & $>$ 600000.000 ms \\

\end{tabular}

\\

\begin{tabular}{lll}

\ref{appendix:sudoku4} Sudoku 4 & Forward Checking & Snelste backtracking(-) \\

Recursions & ? & ? \\

Run Time & $>$ 600000.000 ms & $>$ 600000.000 ms \\

\end{tabular}

\\

\begin{tabular}{lll}

\ref{appendix:sudoku5} Sudoku 5 & Forward Checking & Snelste backtracking(CB) \\

Recursions & 45268 & 416310 \\

Run Time & 24501.0001659 ms & 6559.00001526 ms \\

\end{tabular}

\\

\begin{tabular}{lll}

\ref{appendix:sudoku6} Sudoku 6 & Forward Checking & Snelste backtracking(B) \\

Recursions & 7540 & 235915 \\

Run Time & 3993.00003052 ms & 4964.99991417 \\

\end{tabular}

\\

\begin{tabular}{lll}

\ref{appendix:sudoku7} Sudoku 7 & Forward Checking & Snelste backtracking(B) \\

Recursions & 10903 & 518275 \\

Run Time & 5857.99980164 ms & 12704.9999237 ms \\

\end{tabular}

\\