ASM Sci. J., Special Issue 6, 2019 for SKSM26, 143-150

Hierarchical Simulated Annealing for the Quadratic Assignment Problem

Shah Haziq Shahrin and Mohamed Saifullah Hussin

School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu,
20130 Kuala Nerus, Terengganu, Malaysia

Hybrid methods have been widely used as one of the strategies in stochastic local search (SLS). Simulated Annealing (SA) is one of the SLS method that is commonly used for solving combinatorial problem. In this study, we propose a new hybrid SA that consists of nested SA by performing fast SA and slow SA consistently, known as Hierarchical Simulated Annealing (HSA). To perform HSA, cooling schedule at each level of SA hierarchy needs to be considered. The proposed temperature setting indicates improvement in search space exploration, whether searching around local minima or move to another region. Then, the HSA ideas discussed are tested on the Quadratic Assignment Problem (QAP). The experimental result shows that the performance of HSA is better than SA variants on tested QAP instances, especially on class i instances, due to different search space exploration methods.

Keywords: hybrid stochastic local search; simulated annealing; quadratic assignment problem

I. INTRODUCTION

The idea of Simulated Annealing (SA) algorithm that was proposed by Kirkpatrick *et al.* (1983) originates from Metropolis condition Metropolis *et al.* (1953). To avoid early convergence at the beginning of a search, Metropolis *et al.* (1953) have introduced an efficient diversification strategy. In their study, the probability of accepting a worse solution at the beginning of the search is high, due to the high temperature used. In this way, more neighbors will be accepted at the beginning of a search to avoid early stagnation. According to Eglese (1990), SA can be easily hybridized with another Stochastic Local Search algorithm (SLS) due to the Metropolis condition with strong diversification strategy. He concludes that SA can provide a good initial solution and can improve a solution provided by another SLS. Due to the outstanding performance of SA, recent studies have explored the use of SA as part of hybridization mechanism. In 2006, Pedamallu & Ozdamar proposed a hybrid SA with local search algorithm for constrained optimisation (Misevičius, 2003). They hybridize SA with penalty functions proposed in GAs. In addition, the proposed algorithm is integrated with local search methods proposed by Zhou & Tits (1996). The proposed algorithm has global diversification counter in which the stagnation behaviour can be detected through the search. Other examples are hybrid SA and Tabu Search (Lundy & Mees, 1986), hybrid SA and direct search (Hedar & Fukushima, 2001), hybrid SA and Pattern Search (Hedar & Fukushima, 2004), etc.

In this work, we study the behaviour of the cooling schedule of SA for solving the QAP. Then, we expanded the idea of a cooling schedule by proposing a new hybrid SA. This SA hybrid is based on the idea of the Hierarchical Iterated Local Search (HILS) by Hussin & Stützle (2009). It consists of nested SA by performing fast and slow SA consistently, known as Hierarchical Simulated Annealing (HSA). We examined the cooling schedule at each level of SA hierarchy. Then, the proposed algorithm is tested on QAP instances. The HSA performance is compared to SA variants on selected QAP instances.

II. QUADRATIC ASSIGNMENT PROBLEM

In recent decades, researchers have become increasingly

*Corresponding author's e-mail: shahshahhiha@gmail.com interested in studying the QAP (Burkard, 1998). To date, there remain many unanswered questions about QAP due to its complexity (Sahni & Gonzalez, 1976). Hospital layout problem (Hahn & Krarup, 2001) is a classical QAP example. Given a pair of $n \times n$ matrices, where the first matrix, a_{ij} represents the distance between locations$i$ and j, and the second matrix, b_{kl} represents the amount of flow between facilities k and l. Given the distances between the locations and flows between the facilities, QAP can be defined as the problem of assigning a set of facilities to a set of locations. The objective function of the QAP can be formulated as follows:

$$Min\ \ f(\phi) = \sum_{j=1}^{n} \sum_{i=1}^{n} a_{ij} . b_{\phi(i)\phi(j)}, \qquad (1)$$

where ϕ is an arbitrary permutation of the set of integers $\{1, \dots, n\}$ and $\phi(i)$ is the location of facility i in ϕ. Intuitively, $a_{ij} \cdot b_{\phi(i)\phi(j)}$ represents the cost contribution of simultaneously assigning facility $i$to location $\phi(i)$ and facility j to location $\phi(j)$. The objective of the QAP is to find the lowest total cost contributed by the function (Shahrin & Hussin, 2018).

III. BENCHMARK INSTANCES

In this study, we used four different class problems based on the QAPLIB instances (http://anjos.mgi.polymtl.ca/qaplib/). Class(i) refers to unstructured randomly generated instances. The instances consist of $n_1 \times n_2$ square matrix. The entries in the matrix are generated randomly. Class(ii) refers to instances with grid-based distance matrix. This instance is formed by Manhattan distance on a grid. Class(iii) instances refers to Real-life instances. This instance has many zero and uniformly distributed entries. Class(iv) instances consists of Real-life-like instances. Instances in this class are randomly generated from a uniform distribution.

IV. SIMULATED ANNEALING ALGORITHM

Originally, Simulated Annealing refers to the metal annealing process, from high temperature (melting point) and slowly decreases until equilibrium (solid state). In fact, Kirkpatrick *et al.* (1983) is one of the researchers who started using SA to solve combinatorial optimization problems. Starting from 1984, researchers have studied the implementation of SA for solving the QAP (Misevičius, 2003). Connolly (1990) proposed an improvement of annealing scheme for QAP. The study carried out by Connolly revealed that SA is one of the powerful methods for QAP. Connolly concludes that the performance of SA is optimized at a fixed amount of temperature that need to be considered. Due to the SA structure that is rather simple to build, it has been used for solving many combinatorial problems such as facility layout problem (Allahyari & Azab, 2017) trailer routing problem (Lin *et al.*, 2011), course timetabling problem (Bellio *et al.*, 2016), etc. The basic SA algorithm can be described as follows:

```
Procedure Simulated Annealing
S = RandomInitialSolution
T = Initial Temperature (S)
    repeat
S'= Sequential Neighbor search (S, T)
S = Acceptance Test (S, S', T)
T = Temperature Updated (T)
 If$t_c < t_f$
            T = Temperature Restart (T)
End if
while termination condition not met
End Simulated Annealing
```

Figure 1: Outline of an SA algorithm

First, SA constructs an initial solution by arranging all items at random. The solution will be set as current solution, S. Then, S will be used to calculate the initial temperature, t_c. The final temperature, t_f is set to 1 to ensure that the temperature is converge before restart. The main part of SA, new solutions S' is generated based on current solutions S by performing pairwise exchange. Then, the new solution will be tested whether to be accepted or rejected. This acceptance test is based on Metropolis condition as follows:

$$F(\Delta f) = \begin{cases} 1, & \Delta f \leq 0, \\ e^{-\frac{\Delta f}{t}}, & \Delta f \geq 0, \end{cases} \quad (2)$$

where t is the current temperature and Δf is the difference in objective function value between the current solution S and the new solution S'. Then, the temperature is updated. The temperature will decrease until the minimum temperature is reached and then it is restarted back to the initial temperature. The procedure will be repeated until the termination condition is reached.

V. HIERARCHICAL SIMULATED ANNEALING

In this study, we propose a new SA hybrid to solve the QAP. In some sense, we hybridize an SA with itself (HSA). The idea of HSA originates from Hierarchical Iterated Local Search (HILS) proposed by Hussin & Stützle (2009). We replaced Temperature Restart function in Figure 1 with another SA. The outline of the HSA is shown in Figure 2. We provide a fast SA for inner HSA loop. In this case, the search will focus on a certain region with a smaller search radius. While for the outer loop of HSA, we use a slow SA with high temperatures at the beginning of the search. The search will easily move from one region to another in the search space.

To develop the HSA, temperature setting at each level of SA hierarchy needs to be considered. In fact, temperature is one of the main factors that influence the search process of the algorithm. At present, many temperature setting strategies have been introduced. Each strategy has its own advantage that depends on the structure of the problem to be solved.

```
Procedure Simulated Annealing
$S$ = RandomInitialSolution
$T$ = Initial Temperature ($S$)
    repeat
$S$'= Sequential Neighbor search ($S$, $T$)
$S$= Acceptance Test ($S$, $S$', $T$)
$T$ = Temperature Updated ($T$)
 If$t_c < t_f$
            $S$' = SA ($S$')
End if
while termination condition not met
End Simulated Annealing
```

Figure 2: Outline of HSA algorithm

Two common cooling schedules used in SA: geometric cooling schedules proposed by Kirkpatrick *et al.* (1983),

$$t_{k+1} = \alpha \cdot t_k,\ k = 0, 1, \ldots, It_{max}, \qquad \alpha < 1, \qquad (3)$$

and the one of Lundy and Mees (1986),

$$t_{k+1} = \frac{t_k}{(1 + \beta t_k)}, \qquad (4)$$

$$k = 0, 1, \ldots, It_{max},\ \beta = (t_i - t_f / (M t_i t_f),$$

where It_{max} is the maximum iteration, α is constant parameter, t_i is initial temperature and t_f is final temperature. Then, each temperature is set constant for $c \cdot n$ consecutive swaps, where c is a constant parameter and n is the size of the instance. According to Hussin & Stützle (2009), $c = 100$ is the most appropriate based on several experiments done. The original geometric cooling schedule converges at$t = 0$. With some modifications, SA will converge at any point rather than 0. The modified geometric cooling schedules is as follows:

$$T_k = (T_i - T_f + 1) \cdot (\alpha)^k + T_f - 1, \alpha < 1, \qquad (4)$$

Different temperature setting is used at each HSA level. The initial temperature is calculated as:$t_i = p \cdot \Delta f(Z) Q$, where $\Delta f(Z)$ is the Q difference of the objective function obtained from k random interchange (Q is the third quartile from k, where $k = 100$) and p is a constant parameter. The final temperature is set at $t_f = p \cdot t_i$. At the first level of HSA, we calculate outer t_i by setting $p > 80$. Then, the temperature will decrease slowly. In this situation, each temperature will be remains constant longer (where $c > 80$). Then, we calculate outer t_f by setting $p < 30$. For the second level of HSA, the inner t_i is determined based on the first level of the HSA. In this case, t_f for the first level will be used as t_i for the second level.

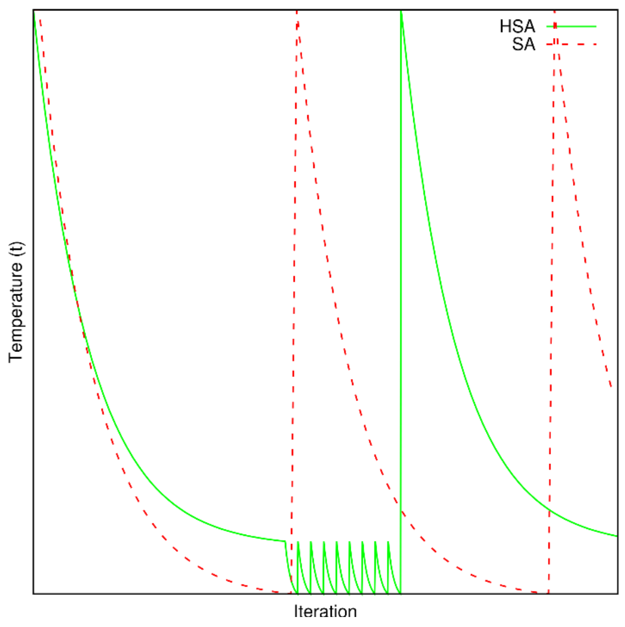

Figure 3. Comparison between SA and HSA annealing schemes

Then, we applied fast cooling schedule, where the temperature decrease is relatively fast ($c < 30$). The inner loops will terminate after $0.3 \cdot n$ iteration. Figure 3 shows the comparison between SA and HSA annealing schemes. We can see that in the same number of iterations, HSA will go through a more frequent temperature changes, thus provides higher chance for improvement by searching for solutions in more confined space.

VI. EXPERIMENTS

In this section, we experimentally evaluate the performance of various SA algorithm on QAP instances. All the experiments have been run on Intel core i3 3.30 GHz quad-core CPU with 4 GB RAM running under classic Ubuntu 16.04. On each instance, a total of 30 trials are conducted and the stopping criteria for the experiment was based on $n \cdot 10^6$ iteration. We set the termination of the algorithm based on iteration due to running multiple tasks at the same CPU.

A. Fast vs Slow Annealing

An experiment was carried out to study the behaviour of different Annealing schemes on QAPLIB instances. Table 1 shows the experimental results of SA with different annealing scheme. The results are recorded in terms of percentage deviation ($\%dev$) to the best known results published in QAPLIB: $\%dev = (A - B) / B \times 100\%$, where A is the objective function obtained from the experiment, while B is the objective function for the best known solution publish from QAPLIB.

We implement SA with various annealing schemes. Each SA follows the procedure in Section 3. Three annealing variants with different initial temperatures and cooling schedules have been studied. For all the SA variants, only two parameters need to be set: the parameter of initial temperature and the parameter of cooling schedule (final temperature is set to 1 for all SA variants). The initial temperature and cooling schedule are calculated based on the formula described in Section 4. We set the first SA variant as SA_s, SA with a slow annealing scheme by setting parameter$p = 1$. Then, each of the temperature will be retained for $c \cdot n$ consecutive swaps, where $c = 100$. The next variant is SA_n (SA with normal annealing scheme), the initial temperature is calculated same as SA_s but using different parameter p, where $p = 0.5$. Then, the parameter in the cooling schedule will be set as $c = 50$. The last one is SA_f, SA with a fast annealing scheme. We set parameters $p = 0.1$ and $c = 10$ as the parameters of t_i and cooling schedule.

Table 1 shows the result of the comparison between SA variants. For the class(i) instances, SA_f shows the best performances among the three SA variants, where it performs well on all tested instances. Followed by SA_s and SA_m where the performance of the two SA variants is close to each other. We can see that, class(i) problem is more appropriate to be solved by using fast annealing. Using a fast cooling schedule, the search process will focus on a smaller region in the search space. SA_s shows the best performance on class(ii) instances, where it performs well on 3 out of 4 tested instances. Then, it is followed by SA_f that performs well on 1 out of 4 and SA_m the worst. In this case, we can assume that the use of different cooling schedules (fast and slow SA) has slightly affect SA performance on class ii instances. For the real-life (class(iii)) instances, SA$_m$ shows the best results, followed by SA$_s$ and SA$_f$ the worst. Class iv instances was generated based on class(iii) instances. Since most of the class iii instances available from QAPLIB are of small size, this might also contribute to the difference in the results obtained. When solving class iv instances with large instances size, more obvious difference can be seen. On class(iv)instances, SA$_s$ is the best where it performs well on 4 out of 4 instances tested. Then, followed by SA$_m$ that performs well on 1 out of 4 instances tested and SA$_f$ is the worst. We can conclude that SA with fast cooling schedule performs well when solving class(i) instances. On class(iii) and (iv), slow cooling schedule should be considered.

Table 1. Experimental results for the comparison between SA variants. The best results are in bold

Instance	Best Known	%dev			Instance	Best Known	%dev		
		SA$_f$	SA$_m$	SA$_s$			SA$_f$	SA$_m$	SA$_s$
tai50a	4938796	**1.448**	*1.921*	1.797	kra30a	88900	*0.268*	**0.000**	**0.000**
tai60a	7205962	**1.453**	*2.069*	1.928	kra30b	91420	**0.000**	**0.000**	0.003
tai80a	13499184	**1.591**	*2.217*	2.016	ste36a	9526	*0.008*	**0.003**	**0.003**
tai100a	21052466	**1.347**	*2.132*	2.007	ste36b	15852	**0.000**	**0.000**	**0.000**
sko72	66256	0.085	*0.118*	**0.078**	tai50b	458821517	*0.699*	0.119	**0.007**
sko82	90998	**0.076**	*0.114*	0.085	tai60b	608215054	*1.064*	**0.001**	**0.001**
sko90	115534	0.116	*0.146*	**0.108**	tai80b	818415043	*0.855*	0.103	**0.055**
sko100a	152002	0.138	*0.147*	**0.114**	tai100b	1185996137	*0.377*	0.086	**0.070**

B. Performance of Hierarchical SA

In this section, we compare the performance of SA and HSA. The implementation of SA and HSA follow the procedure in Section 3 and Section 4. Table 2 and Table 3 shows the experimental results of the comparison. For HSA, we combine fast SA and slow SA based on previous experiments. We provide a slow annealing for outer loop of HSA. In this case, we use a high temperature at the beginning of the search where parameter $p = 1$ (applied on outer t_i). Then, we applied modified geometric coolingschedules according to Formula (4). The parameters in which the temperature will be retained is set to$c = 100$. The algorithm will focus more on diversification rather than intensification. For inner loop of HSA, a fast annealing scheme is provided. We use lower initial temperature where $p = 0.1$ (applied on outer$t_f$and innert_i). and fast cooling schedule wherec $= 10$. In this way, the search process is more focused on smaller regions in the search space. Then, the final temperature for inner SA is set to $t_f = 1$. In this experiment, the proposed HSA is tested using the QAP instances. Instances used in the first experiment (Table 2) are taken from the QAPLIB. Table 2 shows that HSA performs well on class(i), where it shows good performance on all tested instances. When it comes to class(ii) and class(iv), the HSA only works well on a small-sized instance. The HSA shows a good performance on 2 out of 4 tested instances for class ii and 1 out of 4 tested instances for class(iv). While for class(iii) instances, HSA performs well on all instance. Since the size of class iii instances is small, it easy to solve. For the second experiment (Table 3), we used a class of instances used by Hussin & Stützle (2009). Since the instances in QAPLIB are limited, we include this instance set to study the results on instances with different characteristics. The performance of HSA is very impressive. HSA has obtained the lowest average objective function on all instances.

Table 2. Experimental result for the comparison between SA and HSA for QAP instances. The best results are in bold

Instance	Best Known	%dev		Instance	Best Known	%dev	
		SA	HSA			SA	HSA
tai50a	4938796	1.797	**1.365**	kra30a	88900	**0.000**	**0.000**
tai60a	7205962	1.928	**1.463**	kra30b	91420	0.003	**0.000**
tai80a	13499184	2.016	**1.538**	ste36a	9526	0.003	**0.000**
tai100a	21052466	2.007	**1.447**	ste36b	15852	**0.000**	**0.000**
sko72	66256	0.078	**0.068**	tai50b	458821517	0.007	**0.005**
sko82	90998	0.085	**0.078**	tai60b	608215054	**0.001**	0.008
sko90	115534	**0.108**	0.109	tai80b	818415043	**0.055**	0.140
sko100a	152002	**0.114**	0.134	tai100b	1185996137	**0.070**	0.172

Table 3. Experimental result for the comparison between SA and HSA for new instances used by Hussin & Stützle (2009). The best results are in bold

Instance	Best Known	%dev		Instance	Best Known	%dev	
		SA	HSA			SA	HSA
Eu40.00	1862584	1.525	**0.516**	Eu60.00	5032740	2.101	**0.958**
Eu40.33	896208	1.900	**0.281**	Eu60.33	2813754	1.836	**0.789**
Eu40.66	416236	1.308	**0.738**	Eu60.66	1268680	4.668	**1.410**
Eu40.90	79614	0.795	**0.605**	Eu60.90	242722	6.175	**4.616**
Eu50.00	3749244	1.511	**1.501**	Eu80.00	10784164	2.962	**1.546**
Eu50.33	1695648	2.408	**1.490**	Eu80.33	4875086	2.625	**1.198**
Eu50.66	854174	3.353	**1.944**	Eu80.66	2518556	5.301	**3.470**
Eu50.90	196738	8.572	**7.862**	Eu80.90	391560	5.585	**2.859**

VII. CONCLUSION

In this study, we propose a new SA hybrid that consists of nested SA known as Hierarchical Simulated Annealing. Since well-known annealing scheme does not fit in HSA due to the different temperature setting used at each hierarchy level, the new annealing scheme is proposed. Various experiments have been conducted to study the behaviour of annealing schemes proposed. The experiment shows that fast annealing is better than slow annealing on class i QAPLIB instances. However, slow SA shows a better performance than fast SA on class iii and iv QAP instances. We can conclude that annealing scheme should be considered when tackling different QAP classes. An experiment has been run where HSA proposed are compared with SA. the performance of HSA is better than SA on most QAP instances from QAPLIB. In addition, HSA has obtained the lowest average objective function on all instances used by Hussin & Stützle (2009). One of the reasons is that the search process used by SA is less focused on local minimum where the temperature restart is high. This study can be extended by having more levels of hierarchy to solve hard QAP instances. Another option is to use various neighbourhood search strategy such as k-exchange, insertion, inversion, and others on each hierarchy level.

VIII. ACKNOWLEDGEMENT

This work was supported by the Research Management & Innovation Centre (RMIC) of UMT, funded by Fundamental Research Grant Scheme (FRGS), Ministry of Education, Malaysia. Mohamed Saifullah Hussin acknowledges support from Universiti Malaysia Terengganu.

IX. REFERENCES

Allahyari, M. Z., & Azab, A. (2017). Facility Layout Problem for Cellular Manufacturing Systems. In *Computational Optimization in Engineering-Paradigms and Applications*. InTech.

Bellio, R., Ceschia, S., Di Gaspero, L., Schaerf, A., & Urli, T. (2016). Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. *Computers & Operations Research*, *65*, 83-92.

Burkard, R. E., Çela, E., Pardalos, P. M., & Pitsoulis, L. S. (1998). The quadratic assignment problem. In *Handbook of combinatorial optimization* (pp. 1713-1809). Springer, Boston, MA.

Connolly, D. T. (1990). An improved annealing scheme for the QAP. *European Journal of Operational Research*, *46*(1), 93-100.

Eglese, R. W. (1990). Simulated annealing: a tool for operational research. *European journal of operational research*, *46*(3), 271-281.

Hahn, P. M., & Krarup, J. (2001). A hospital facility layout problem finally solved. *Journal of Intelligent Manufacturing*, *12*(5-6), 487-496.

Hedar, A., & Fukushima, M. (2001). Hybrid simulated annealing and direct search method for nonlinear global optimization. *Department of Applied Mathematics & Physics Kyoto University*, 2001-2013.

Hedar, A. R., & Fukushima, M. (2004). Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. *Optimization Methods and Software*, *19*(3-4), 291-308.

Hussin, M. S., & Stützle, T. (2009, October). Hierarchical iterated local search for the quadratic assignment problem. In *International Workshop on Hybrid Metaheuristics* (pp. 115-129). Springer, Berlin, Heidelberg.

Hussin, M. S., & Stützle, T. (2014). Tabu search vs. simulated annealing as a function of the size of quadratic assignment problem instances. *Computers & operations research*, *43*, 286-291.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671-680.

Krarup, J., & Pruzan, P. M. (1978). Computer-aided layout design. *Mathematical programming in use*, 75-94.

Lin, S. W., Vincent, F. Y., & Lu, C. C. (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Systems with Applications*, *38*(12), 15244-15252.

Lundy, M., & Mees, A. (1986). Convergence of an annealing algorithm. *Mathematical programming*, *34*(1), 111-124.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, *21*(6), 1087-1092.

Misevičius, A. (2003). A modified simulated annealing algorithm for the quadratic assignment problem. *Informatica*, *14*(4), 497-514.

Pedamallu, C. S., &Ozdamar, L. (2008). Investigating a hybrid simulated annealing and local search algorithm for constrained optimization. *European Journal of Operational Research*, *185*(3), 1230-1245.

Sahni, S., & Gonzalez, T. (1976). P-complete approximation problems. *Journal of the ACM (JACM)*, *23*(3), 555-565.

Shahrin, S. H., & Hussin, M. S. (2018). Comparisons of simulated annealing temperature schedule based on QAPLIB instances. In *AIP Conference Proceedings* (Vol. 1974, No. 1, p. 020091). AIP Publishing.

Skorin-Kapov, J. (1990). Tabu search applied to the quadratic assignment problem. *ORSA Journal on computing*, *2*(1), 33-45.

Steinberg, L. (1961). The backboard wiring problem: A placement algorithm. *Siam Review*, *3*(1), 37-50

Taillard, É. D. (1995). Comparison of iterative searches for the quadratic assignment problem. Location science, 3(2), 87-105.

Zhou, J. L., & Tits, A. L. (1996). An SQP algorithm for finely discretized continuous minimax problems and other minimax problems with many objective functions. *SIAM Journal on Optimization*, *6*(2), 461-4.