ECAI 2008
M. Ghallab et al. (Eds.)
IOS Press, 2008
© 2008 The authors and IOS Press. All rights reserved.
doi:10.3233/978-1-58603-891-5-857

Simulated Annealing for Coalition Formation

Helena Keinänen[1] and **Misa Keinänen**[2]

1 INTRODUCTION

We study coalition formation in characteristic function games (CFGs) [4, 5]. Consider a n-person cooperative game where $A = \{a_1, a_2, \dots, a_n\}$ is the set of agents. A coalition is any non-empty subset S of A, i.e. $S \subseteq A$ such that $S \neq \emptyset$. In CFGs a characteristic function $v : S \to \mathbb{R}$ assigns real values (worths) to coalitions such that the function may be incomplete. A coalition structure CS is a partition of A into mutually disjoint coalitions in which, for all S, $S' \subseteq A$, we have $S \cap S' = \emptyset$, $S \neq S'$ and $\bigcup_{S \in CS} = A$. The value of a coalition structure $V(CS)$ is called social welfare, and it is defined as $V(CS) = \sum_{S \in CS} v(S)$. Given a set of agents $A = \{a_1, a_2, \dots, a_n\}$ together with a characteristic function $v : S \to \mathbb{R}$, our aim is to find a coalition structure with maximum social welfare. It is shown in [5] that finding a social welfare maximizing coalition structure is a NP-complete problem, and that the number of coalition structures is $O(n^n)$ and $\omega(n^{n/2})$.

Motivated by the observations in [6, 7] that genetic algorithms provide a useful tool for searching the maximal sum of the values of coalitions, we show that simulated annealing (SA) [1, 3] provides also a very competitive approach to the problem. We observe that the SA algorithm with a suitable neighbourhood relation often finds better values or even the optimal coalition structures well before the state-of-the-art algorithms in [2, 4, 5].

2 SA FOR COALITION FORMATION

Algorithm 1 shows our SA algorithm for optimizing the social welfare of a CFG. The algorithm takes a characteristic function $v : S \to \mathbb{R}$ for an n-agent CFG as its input. Additional inputs are iteration limit c_{max}, initial temperature t_{init}, and the cooling ratio α. c is to keep track of the number of iterations. CS_{best} records the coalition structure with the highest social welfare among the ones seen. At each iteration a random neighbour solution CS' of coalition structure CS is picked according to a specific neighbourhood $Neighbour()$. The search proceeds with an adjacent coalition structure CS' of the original coalition structure CS, if CS' yields a better social welfare than CS. Otherwise, the search is continued with CS' with probability $e^{(V(CS')-V(CS))/t}$. The temperature t decreases after each iteration according to an annealing schedule $t = \alpha t$ where $0 < \alpha < 1$.

The performances of SA algorithms are very sensitive to parameter adjustments as well as to neighbourhood selection. Given a set of agents A together with a characteristic function v, let M denote the set of all coalition structures that can be formed. The neighbourhood is a function $Neighbour : M \to 2^M \setminus \emptyset$ which maps coalition structures to the sets of their neighbour coalition structures.

We found out that the following two neighbourhoods are particularly appropriate for Alg. 1. **Split/merge** neighbourhood, in which $CS' \in Neighbour(CS)$ if and only if CS' can be obtained from CS by either (i) splitting one coalition in CS into two disjoint coalitions in CS', or (ii) merging two distinct coalitions of CS into a single coalition in CS'. **Shift** neighbourhood, in which $CS' \in Neighbour(CS)$ if and only if CS' can be obtained from CS by shifting exactly one agent from a coalition to another coalition.

Algorithm 1

```
Inputs: c_max, t_init, alpha
External: V()
c = 0;
t = t_init;
CS = random initial coalition structure;
CS_best = CS;
while c < c_max do
  CS' = random neighbour of CS in Neighbour(CS);
  if V(CS') > V(CS) then
    CS = CS';
    if V(CS)>V(CS_best) then CS_best=CS;
  else
    with probability e^((V(CS')-V(CS))/t)
      CS = CS';
  c = c+1;
  t = alpha*t;
return CS_best;
```

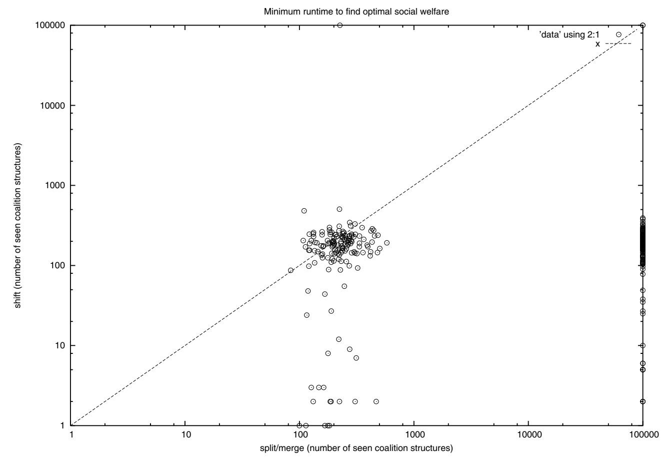
3 EXPERIMENTAL RESULTS

We have implemented the Alg. 1 in C, and evaluated its performance on CFG problems. As our benchmarks we use problems from [2, 4, 5, 6]. In the following we present experimental results considering in particular solution quality, robustness, and runtime performances of various algorihms.

The Fig. 1 (left) shows a robustness comparison of split/merge and shift neighbourhoods for the SA algorithm on 300 randomly generated 10-agent CFG problem instances. For each problem instance, an incomplete characteristic function v was generated to assign random coalition values between $[0, 1]$. An exhaustive search was first used to find social welfare maximizing coalition structures, and then the SA algorithm was used to find optimal solutions in the following way. For both neighbourhoods, we executed 11 runs on every problem instance with approximately optimal parameters $\alpha = 0.99$ and $t_{init} = 1$. The runtime limit for each run was set to 100000 coalition structures. We plot the minimum execution times of 11 runs to find an optimal social welfare. The shift neighbourhood is much more robust than the split/merge. SA with shift neighbourhood is able to find the optimum solution in 298 of the 300 instances. In contrast, SA with the split/merge times out in 136 instances without finding an optimum solution. Tne SA with the shift neighbourhood is mostly able to find the optimum values with substantially fewer search steps than SA with the split/merge. Irrespective of the parameter variation the behaviour of the shift neighbourhood was superior.

To compare the solution qualities of SA with the two different neighbourhoods, we investigate the behaviours on 100 randomly

[1] Helsinki University of Technology, Finland, helena.keinanen@tkk.fi
[2] Sampo Life Insurance Company Ltd. , Finland, misa.keinanen@gmail.com



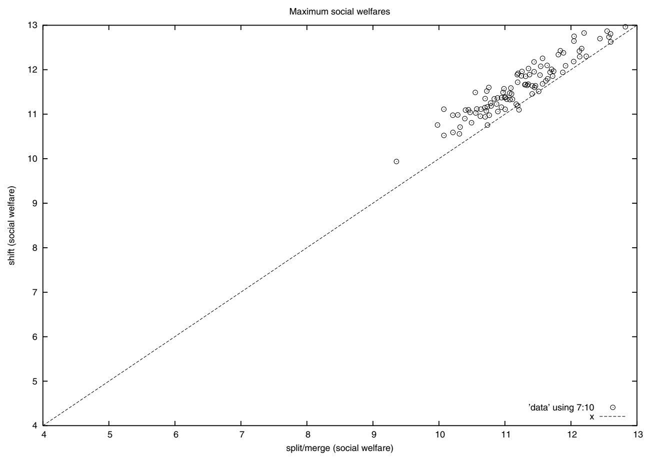


Figure 1. Comparisons of neighbourhood relations on random CFGs.

generated 20-agent CFG problem instances, again random coalition values in $[0,1]$. Fig. 1 (right) shows the correlation between the solution quality of SA with the split/merge neighbourhood and SA with the shift neighbourhood. The plot illustrates the maximum social welfares found, measured from 11 runs per neighbourhood. The runtime limit was set to $2^{20-1} = 524288$ coalition structures, and we used the approximately optimal annealing schedule where $\alpha = 0.99$ and $t_{init} = 1$. These results clearly show that SA with the shift neighbourhood outperforms SA with the split/merge neighbourhood.

We have also implemented in C algorithms presented in [2, 4, 5], and a random search on the graph induced by the neighbourhood relations. We compared the performances of SA, Random search and the anytime algorithms on a set of randomly generated 10-agent CFGs with coalition's values picked randomly from a uniform distribution $[0,1]$. Fig. 2 shows the cumulative solution qualities over runtime (measured as seen coalition structures), on a representative problem instance. The initial temperature for SA is $t_{init} = 1$ and α is fixed to 0.8. Both SA and Random search are run only once. The SA algorithm finds good solutions very quickly. The SA with the shift neighbourhood finds the optimum within short runtime, and also SA with the split/merge neighbourhood climbs very close to the optimum. Random search with both neighbourhoods manages to find quickly relatively good solutions. However, as SA with the split/merge, Random search do not find any maximal social welfare. The anytime algorithm searches for a long time without finding good solutions, but then finally sees a coalition structure with maximal social welfare.

Finally, we conducted further experiments on 100 random 20-agent CFGs with coalition's values in $[0,1]$. For each problem instance, we collected the minimal, median and maximal social welfares measured from 11 runs per algorithm. In these tests, the runtime limit for all algorithms was set to $2^{20-1} = 524288$ coalition structures. We used the SA with the shift neighbourhood, and the SA parameters were the approximately optimal $\alpha = 0.99$ and $t_{init} = 1$. The results are consistent with the results of the previous experiments. For all problem instances the SA algorithm substantially outperforms the anytime algorithms in [2, 4, 5]. Notably, every social welfare found with the anytime algorithms [2, 4, 5] is smaller than 2, whereas the SA always finds social welfares better than 9. The results with the SA provide an improvement in the order of a factor 5.

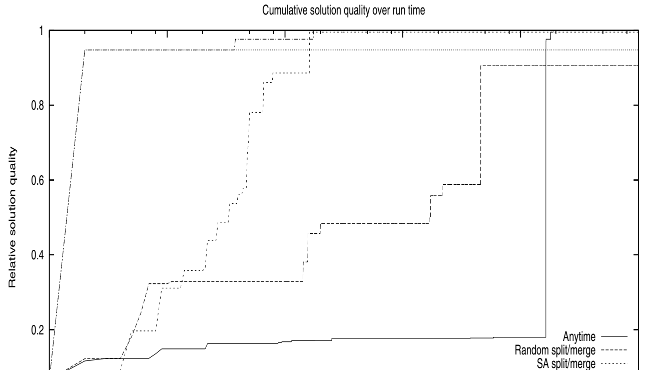


Figure 2. A comparison of SA, Random search and Anytime algorithms.

REFERENCES

[1] V. Černý, 'Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm', *J. of Optimization Theory and Applications*, **45**, 41–51, (1985).

[2] V.D. Dang and N.R. Jennings, 'Generating coalition structures with finite bound from the optimal guarantees', in *Proc. 3rd Int. Conf. on Autonomous Agents and Multi-Agent Systems*, 546–571, 2004.

[3] S. Kirkpatrick, C.D. Gelatt, Jr. and M.P. Vecchi, 'Optimization by simulated annealing', *Science*, **220**, 671–680, (1983).

[4] K.S. Larson and T.W. Sandholm, 'Anytime coalition structure generation: An average case study', *J. Expt. Theor. Artif. Intell.*, **12**, 23–42, (2000).

[5] T. Sandholm, K. Larson, M. Andersson, O. Shehory and F. Tohmé, 'Coalition structure generation with worst case guarantees', *Artificial Intelligence*, **111**, 209–239, (1999).

[6] S. Sen and P.S. Dutta, 'Searching for optimal coalition structures', in *Proc. 4th Int. Conf. on Multi-agent Systems*, 286–292, 2000.

[7] J. Yang and Z. Luo, 'Coalition formation mechanism in multi-agent systems based on genetic algorithms', *Applied Soft Computing*, **7**, 561–568, (2007).