A Decentralized Heuristic for Multiple-Choice Combinatorial Optimization Problems

Christian Hinrichs, Sebastian Lehnhoff and Michael Sonnenschein

1 Introduction

In 0-1 multiple-choice combinatorial optimization problems, multiple sets or classes of elements are given, from which each exactly one element has to be chosen to form a solution. The goal is to find a solution that minimizes (or maximizes) a given objective function. Such problems are typically NP-hard in the weak sense, so that they can be solved in polynomial time using dynamic programming [4]. By exploiting the specific structure of a given problem, even linear time can be achieved. For example, the well-known multiple-choice knapsack problem can easily be solved using the *core* concept [6]. But such methods cannot be applied to all problems of this class for two reasons: First, a core may be difficult to find, as it is the case in some (multiple-choice) subset-sum problems [7]. Second, and more importantly, the problem may have to be solved in a decentralized way, so that global knowledge is not available. This is especially the case in distributed systems with autonomous actors. For example, such a system may not support gathering global knowledge due to its openness, or the cost for doing this may be very high. Also, the collection of global knowledge at certain stages of an algorithm (i.e. at startup) leads to synchronization points, which are undesirable in many cases. Finally, such actions may violate privacy considerations. Typical application fields for these kinds of problems are distributed resource allocation, wireless sensor networks, media streaming in bandwith-contrained environments, logistics and decentralized energy management.

C. Hinrichs (✉) · M. Sonnenschein
University of Oldenburg, Oldenburg, Germany
e-mail: hinrichs@informatik.uni-oldenburg.de

M. Sonnenschein
e-mail: sonnenschein@informatik.uni-oldenburg.de

S. Lehnhoff
OFFIS, Institute for Information Technology, Oldenburg, Germany
e-mail: lehnhoff@offis.de

S. Helber et al. (eds.), *Operations Research Proceedings 2012*,
Operations Research Proceedings, DOI: 10.1007/978-3-319-00795-3_43,

In this contribution, we present a **C**ombinatorial **O**ptimization **H**euristic for **D**istributed **A**gents (COHDA). Unlike other population-based heuristics (i.e. particle swarm optimization), the individuals ("agents") in COHDA do not describe candidate solutions. Instead, each agent represents a class of elements of the considered multiple-choice combinatorial opimization problem, so that solutions comprise individual decisions from all participating agents. The heuristic solves the given problem in a completely decentralized manner, relying on individual knowledge and utilizing cooperative behaviour. The individual knowledge bases are formed through local perception plus beliefs about (possibly incomplete or outdated) global knowledge. Convergence of the search process is achieved by distributing the beliefs to other agents. This kind of heuristic corresponds to the *Cooperative Algorithmic-Level Parallel Model* defined in [9].

2 Problem Definition and Model

As stated in the introduction, multiple-choice subset-sum problems (MC-SSP) belong to the rather difficult types among combinatorial optimization problems, because their structure is less exploitable. Hence, in this contribution, we focus on this problem type. We are given m classes with each class i containing n_i elements. The jth element of class i has weight w_{ij}. From each class, exactly one element has to be chosen for a feasible solution. In MC-SSP, there exists a capacity c which defines an upper bound that should be approximated, but not exceeded by the sum of the weights of the chosen elements. We generalize from this formulation by removing the upper bound constraint. The goal is therefore to approximate c as close as possible from any side, which yields a problem with increased solution space. We call this generalization *multiple-choice combinatorial optimization problem* (MC-COP). Formally, it can be expressed with an integer programming model:

$$\text{(MC-COP)} \qquad \min \; d\left(c, \sum_{i=1}^{m}\sum_{j=1}^{n_i}\left(w_{ij}\cdot x_{ij}\right)\right) \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{n_i} x_{ij} = 1, \;\; i = 1\ldots m,$$

$$x_{ij} \in \{0,1\}, \;\; i = 1\ldots m, \;\; j = 1\ldots n.$$

The objective function d can be defined arbitrarily, i.e. the 1-norm might be used:

$$d(u,v) = \|u - v\|_1 \tag{2}$$

Since we are targeting distributed systems, we mapped this model to a multi-agent system (MAS). In this MAS, each agent represents an element class. Hence, each agent a_i has to select one of its elements to be included in the solution by assigning $x_{ij} = 1$, where j is the index of the selected element. The difficulty of the problem arises from the lack of global knowledge at agent level, i.e. there is no global decision maker, and the solution has to be determined in a decentralized fashion. In order to accomplish that, for each agent a_i, a neighborhood set $\mathcal{N}_i$ of other agents is defined with whom a_i is able to communicate. The communication network may be expressed with an undirected graph. An important property of the communication layer are message delays: We assume that sent messages take an arbitrary time until they are received by the target agent, as it is the case in most real communication networks like the internet. However, we simplify from reality by limiting this delay to a known upper bound $t_{msg,max}$, and we assume that no messages are lost during transmission. Finally, we presume a central operator, who is able to broadcast the target value c to all agents and thus can initiate the heuristic.

3 Heuristic

The task of each agent is to select one of its own controlled elements, so that the sum of the weights of all selected elements in the population minimizes the objective function d. To accomplish that, each agent communicates with its neighbors and shares knowledge that helps in selecting optimal elements. In our approach, an agent a_i needs to exchange only two items with its neighbors: The best combination of weights $\hat{W}_{i,best}$ of selected elements that the agent has seen during the process so far and the currently selected weights $\hat{W}_{i,current}$ that the agent is aware of. Each of these communicated weights is labelled, where a label $l\left(w_{ij}\right) = (i,\ s_i)$ contains the unique ID i of the agent the selected weight belongs to, and a counter s_i that reflects the "age" of the selection. Note that, however $\hat{W}_{i,best}$ and $\hat{W}_{i,current}$ must each satisfy the constraints of MC-COP (i.e. they must not contain more than one chosen weight per agent, see (1)), these sets are allowed to be incomplete. At the beginning of the process, $\hat{W}_{i,best}$ and $\hat{W}_{i,current}$ will each contain only the weight $w_{i,init}$ of the initially selected element of the agent a_i itself, labelled $l\left(w_{i,init}\right) = (i,\ 0)$. However, as the agent begins to communicate with its neighbors, the sets will be updated with each information exchange. The agent is allowed to change its element selection with the help of the knowledge contained in $\hat{W}_{i,best}$ and $\hat{W}_{i,current}$ at any time under one condition: It has to make this selection public to its neighbors, while labelling it with its age counter increased by one. Hence, the age counter s_i of an agent a_i increases each time the agent changes its selection, thus making it possible to decide if received information concerning an agent is newer than an already stored value. The complete heuristic executed by each agent comprises three steps and can be described as follows.

1. **(update)** Every time an agent a_i receives a message, its local knowledge base is updated with the information received. This can be either the target value c, which starts the process and is broadcasted by the central operator, or information from a neighbor agent a_h, containing the sets $\hat{W}_{h,best}$ and $\hat{W}_{h,current}$. In the first case, the value c is stored locally, and step 3 is executed without further conditions. Otherwise, the local knowledge base $\hat{W}_{i,current}$ is updated with the items in $\hat{W}_{h,current}$ by adding selected weights from agents not known so far, and replacing outdated selections according to the age counters in the associated labels. The local set $\hat{W}_{i,best}$ is replaced by $\hat{W}_{h,best}$ if the latter contains more elements, or if it yields a better value using the objective function. If either $\hat{W}_{i,best}$ or $\hat{W}_{i,current}$ has been changed during this process, step 3 is executed afterwards.
2. **(choose**) In this step, the agent iterates through its own weights and calculates the value of the objective function for each of these weights when combined with the items in $\hat{W}_{i,current}$, thus testing which selection $w_{i,new}$ fits the best into the currently existing configuration. Note that this local view on the system defined by $\hat{W}_{i,current}$ will most likely already be outdated due to the asynchronous execution of the heuristic in each agent. Yet it describes an approximation to the global system state that helps in finding optimal selections. The best resulting objective value found in the iteration of weights is then compared to the objective value of $\hat{W}_{i,best}$. If the found objective value of $\hat{W}_{i,current} \cup \{w_{i,new}\}$ is better than the value of $\hat{W}_{i,best}$ and the size of $\hat{W}_{i,current} \cup \{w_{i,new}\}$ is at least as large as the size of $\hat{W}_{i,best}$, the weight $w_{i,new}$ is marked as selected and the new best known configuration is stored as $\hat{W}_{i,best} := \hat{W}_{i,current} \cup \{w_{i,new}\}$. Otherwise, the agent reverts its selection to the one stored in $\hat{W}_{i,best}$.
   If the selected weight of agent a_i has changed during this process, step 3 is executed afterwards.
3. **(ublish)** In this step, the agent sends its currently stored sets $\hat{W}_{i,best}$ and $\hat{W}_{i,current}$ to all of its neighbors.

Utilizing these actions, the agents will update each other iteratively with the newest information they are each aware of, while simultaneously trying to find the best global combination of weights. This leads to some important properties of COHDA:

Convergence. The conditions in step 3 will first cause the individual sets $\hat{W}_{i,best}$ to increase in size until completeness, hence spreading knowledge in the system, even between agents that are not directly connected. Afterwards, the heuristic will start to converge, because only better rated configurations survive the selection process in each iteration. When no agent is able to find a better configuration, all agents will eventually stick to a common best known configuration.

Cooperation. The agents need to cooperate with each other, i.e., they need to know the global objective function d, and have to be trustworthy in their exchanged messages.

Termination. After convergence, no agent will change its selected element any more. Hence, termination of the heuristic can be determined by observing inactivity of all participating agents. From the point of view of the global observer, this is the

case when no messages have been sent for a duration $t_{msg,max}$ by any agent (thus eliminating the uncertainty introduced by arbitrary message delays).

Completeness. Due to the lack of global knowledge, the algorithm is not complete; it cannot be guaranteed that the optimum is found.

4 Evaluation

We implemented COHDA in a discrete-event simulation system that simulates a communication network using integer-valued message delay time steps. Each simulation step represents one simulated time step. The evaluation example corresponds to an application from the domain of decentralized energy management. In the electric grid, the supply and demand of energy has to be balanced at every point in time. In the context of operations research, this problem has been formulated as the electrical generation unit commitment problem (UC, see [5] for a comprehensive survey). In our evaluation, we do not only consider generators, but include flexible loads as well, which is known as demand side management (DSM). We mapped this problem to MC-COP by replacing the single-valued integer variables in (1) by q-dimensional vectors and allowed negative values, thus being able to model a time series of electrical load (with k discrete time steps, $k = 1 \ldots q$), which should be approximated by the sum of the agent's individual load curves; see [2] for a more detailed formalization. The considered problem instances were generated using fixed parameters $m = 30$ (number of element classes), 100 different elements in every class with $q = 96$ dimensions each, and strongly correlated weight values w_{ij}, which have been identified as hard instances for SSP-like problems [3, 6, 7]. Five different target time series c_h were generated as described in [1], thus $1 \leq h \leq 5$. Each of these five configurations was simulated 100 times with a set of 100 seeds for the random number generator, so that within each configuration the same 100 seeds were used for the 100 simulation runs. The communication network was modelled after the small world paradigm [8] with a rewiring propability $\Phi = 2.0$ and message delays uniformly distributed in $[1,\ 10]$. The output of the objective function was normalized to $[0,\ 1]$ using an optimal-to-worst-interval according to [9].

Table 1 COHDA evaluation results: final objective value d, number of messages per agent per time step msg, number of time steps until termination t for different targets c_h

	$h = 1$		$h = 2$		$h = 3$		$h = 4$		$h = 5$	
	mean	std	mean	std	mean	std	mean	std	mean	std
d	0.0044	0.0122	0.0024	0.0084	0.0031	0.0142	0.0031	0.0096	0.0027	0.0088
msg	0.5495	0.0201	0.5400	0.0222	0.5354	0.0234	0.5456	0.0279	0.5477	0.0255
t	373.59	171.66	465.41	216.07	526.43	236.98	443.58	226.83	408.16	210.93

Summarized in Table 1 are the means and standard deviations of the final objective value d, the number of messages sent per agent per time step (msg), and the number of time steps t until termination for each of the five configurations.

5 Conclusion

The evaluation results show that COHDA is able to find near-optimal solutions within a range of 0.0031 ± 0.0106 (mean over all configurations) of the theoretical optimal value $d_{opt} = 0$ for the given multiple-choice combinatorial optimization problem. While the run-times vary rather large (443.434 ± 212.493 simulated time steps), the exchanged messages remain very constant at 0.5436 ± 0.0238 messages per agent per time step on average over all simulation runs. We will publish a more thorough evaluation in a subsequent paper. There, the approach will be compared to related work, and a multi-objective variant of COHDA will be presented.

Future work will mainly focus on adaptivity. In this context, we will extend MCCOP with a dynamic objective function, and we will incorporate the ability to handle non-constant agent populations as well as adaptive communication networks into COHDA. An interesting research topic resulting from these extensions is the stability of the heuristic against destructive/harmful agents.

References

1. Han, B., Leblet, J., Simon, G.: Hard multidimensional multiple choice knapsack problems, an empirical study. Computers & Operations Research **37**(1), 172–181 (2010). doi:10.1016/j.cor.2009.04.006
2. Hinrichs, C., Vogel, U., Sonnenschein, M.: Approaching Decentralized Demand Side Management via Self-Organizing Agents. In: Yolum, Tumer, Stone, Sonenberg (eds.) ATES Workshop, Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011). Taipei, Taiwan (2011).
3. Lust, T., Teghem, J.: The multiobjective multidimensional knapsack problem: a survey and a new approach. International Transactions in Operational Research 19(4), 495–520 (2012). doi:10.1111/j.1475-3995.2011.00840.x
4. Martello, S., Toth, P.: Knapsack problems, 1 edn. John Wiley & Sons (1990).
5. Padhy, N.: Unit Commitment-A Bibliographical Survey. IEEE Transactions on Power Systems **19**(2), 1196–1205 (2004). doi:10.1109/TPWRS.2003.821611
6. Pisinger, D.: A minimal algorithm for the multiple-choice knapsack problem. European Journal of Operational Research 83(2), 394–410 (1995).doi:10.1016/0377-2217(95)00015-I
7. Pisinger, D.: Linear Time Algorithms for Knapsack Problems with Bounded Weights. Journal of Algorithms **33**(1), 1–14 (1999). doi:10.1006/jagm.1999.1034
8. Strogatz, S.H.: Exploring complex networks. Nature **410**(6825), 268–276 (2001). doi:10.1038/35065725
9. Talbi, E.G.: Metaheuristics. John Wiley & Sons, Inc., Hoboken, NJ, USA (2009). 10.1002/9780470496916.