An Improved Hybrid Genetic Algorithm for the Generalized Assignment Problem [*]

Harald Feltl and Günther R. Raidl
Institute of Computer Graphics and Algorithms
Vienna University of Technology, Vienna, Austria
harald.feltl@wave-solutions.com, raidl@ads.tuwien.ac.at

ABSTRACT

We consider the generalized assignment problem in which the objective is to find a minimum cost assignment of a set of jobs to a set of agents subject to resource constraints. The presented new approach is based on a previously published, successful hybrid genetic algorithm and includes as new features two alternative initialization heuristics, a modified selection and replacement scheme for handling infeasible solutions more appropriately, and a heuristic mutation operator. Tests are performed on standard test instances from the literature and on newly created, larger and more difficult instances. The presented genetic algorithm with its two initialization variants is compared to the previous genetic algorithm and to the commercial general purpose branch-and-cut system CPLEX. Results indicate that CPLEX is able to solve relatively large instances of the general assignment problem to provable optimality. For the largest and most difficult instances, however, the proposed genetic algorithm yields on average the best results in shortest time.

Keywords

Generalized assignment problem, hybrid genetic algorithm, linear programming

1. INTRODUCTION

In the *generalized assignment problem* (GAP), the objective is to find a minimum costs assignment of $n > 0$ jobs to $m > 0$ agents such that each job is assigned to exactly one agent, subject to agents' available capacities. Hereby, each job/agent assignment may have individual costs and resource requirements. Several applications of this problem exist in areas like computer and communication networks, location problems, vehicle routing, and machine scheduling. The GAP is known to be strongly NP-hard [6]. Related classical problems are the multiple knapsack, the bin packing, and the set partitioning problem.

[*]This work is supported by the Austrian Science Fund (FWF) under grant P16263-N04.

More formally, the generalized assignment problem can be stated as follows. Let $I = \{1, \ldots, m\}$ be the set of agents and $J = \{1, \ldots, n\}$ the set of jobs. For each agent i we are given a resource capacity $b_i > 0$. For each $i \in I$ and each $j \in J$ we are given costs $c_{i,j} > 0$ and resource requirements $r_{i,j} > 0$ for assigning job j to agent i.

The objective is to find an assignment matrix $x = (x_{i,j})$, with $x_{i,j} = 1$ if agent i performs job j and 0 otherwise, which minimizes the total costs

$$c(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j} \cdot x_{i,j} \qquad (1)$$

subject to

$$\sum_{j=1}^{n} r_{i,j} \cdot x_{i,j} \leq b_i, \quad \forall i \in I, \qquad (2)$$

$$\sum_{i=1}^{m} x_{i,j} = 1, \quad \forall j \in J, \qquad (3)$$

$$x_{i,j} \in \{0, 1\}, \quad \forall i \in I, \ \forall j \in J. \qquad (4)$$

The *capacity constraints* (2) ensure that the total resource requirement of the jobs assigned to each agent do not exceed its capacity. The *assignment constraints* (3) guarantee that each job is assigned to exactly one agent.

The GAP has already been approached by several researchers with different optimization techniques. The next section will provide a brief summary of previous approaches. Among them is also a successful hybrid genetic algorithm by Chu and Beasley [4], which forms the basis of the new algorithm proposed in this article. Section 3 describes two new heuristics for creating promising initial candidate solutions, an alternative selection and replacement strategy that treats infeasible candidate solutions in a special way, and a new heuristic mutation operator. In Section 4, results of empirical tests on standard instances and newly created larger instances are documented. The new approach with its two initialization variants is compared to Chu and Beasley's genetic algorithm and the commercial branch-and-cut integer linear programming solver CPLEX. After the discussion of the results, Section 5 concludes this article.

2. PREVIOUS WORK

The existing approaches for the GAP can be divided into exact and heuristic methods. Most exact methods are based on branch-and-bound and also use heuristics to start with good initial solutions. The following paragraphs describe two branch-and-bound approaches, which differ significantly in the way the bounds are calculated, and the genetic algorithm from Chu and Beasley. An extensive survey on previous approaches for the GAP can be found in [3].

Martello and Toth [9] proposed an algorithm which starts with the following heuristic—denoted by MTH—in order to obtain a good initial solution. Let $\mu_{i,j}$ be a heuristic measure of the desirability of assigning job j to agent i. MTH iteratively considers all the unassigned jobs and determines a job j^* with maximum difference between the largest and the second largest $\mu_{i,j}$. Hereby, only valid agents are considered, i.e. the capacity constraints may not be violated. Job j^* is then assigned to the agent i with the largest μ_{i,j^*}. In the second part of the heuristic, the current solution is eventually improved by a local search procedure. So-called *shift* operations are applied until the solution cannot further be improved. The shift operation considers each job and tries to reassign it to another feasible agent with smaller costs $c_{i,j}$. Four different desirability measures are applied: (a) $\mu_{i,j} = -c_{i,j}$, (b) $\mu_{i,j} = -c_{i,j}/r_{i,j}$, (c) $\mu_{i,j} = -r_{i,j}$, and (d) $\mu_{i,j} = -r_{i,j}/b'_i$, where b'_i is the remaining capacity of agent i. The best solution obtained after trying each of them is the final MTH solution. Note, however, that this heuristic is not guaranteed to terminate with a feasible solution, since the algorithm may run out of feasible agents for a job in its first part.

If a feasible heuristic solution is found, it is used as an initial global upper bound for the following depth-first branch-and-bound scheme. At each node of the decision tree, a lower bound is obtained by solving the relaxed problem (1), (2), and (4). Thus, the assignment constraints (3) are ignored. This subproblem corresponds to m individual 0–1 single knapsack problems, which can be efficiently solved by dynamic programming [8]. If the solution of the relaxed problem satisfies constraints (3), the node generates no descendants; otherwise the infeasibility can be of two types: (a) a job j is not assigned, i.e. $\sum_{i \in I} x_{i,j} = 0$, or (b) a job j is assigned more than once, i.e. $\sum_{i \in I} x_{i,j} > 1$. At every node of the branch-and-bound tree a reduction phase is used to further reduce the size of the search space.

Savelsbergh [11] described an approach that employs column generation together with branch-and-bound to obtain optimal solutions to a set partitioning formulation of the GAP. Column generation is used to solve the linear programming relaxation of the disaggregated formulation for the GAP. Branching is performed if the solution to the linear programming relaxation is not feasible for the integer program. To further reduce the size of the branch-and-bound tree, a heuristic similar to MTH is used and a surrogate relaxation of the capacity constraints is performed as proposed by Jörnsten and Näsberg [7].

The mentioned branch-and-bound algorithms are only effective on certain GAP instances of small and medium size. Larger and more complex instances are tackled by applying meta-heuristics to obtain approximate solutions.

Osman [10] developed a hybrid of simulated annealing and tabu search, and experimented with several variants of local search descent methods.

Amini and Racer [1] described a variable-depth neighborhood search heuristic based on a two-phase local search descent method. Phase one generates an initial solution. Phase two consists of a doubly-nested refinement procedure in which either a feasible shift of a job from one agent to another or a feasible swap of two jobs is performed.

To our knowledge, Jörnsten and Värbrand [7] developed the first genetic algorithm (GA) for the GAP using a pivot and complement heuristic.

Algorithm 1 Heuristic Improvement (S)

```
 1: compute accumulated resource requirements:
    $R_i = \sum_{j \in J | S_j = i} r_{i,j}, \ \forall i \in I$;
 2: /* Phase 1: improve feasibility */
 3: for $i = 1$ to $m$ do
 4:   if $R_i > b_i$ then
 5:     $T \leftarrow \{j \in J \mid S_j = i\}$;
 6:     repeat
 7:       randomly select a $j \in T$; $T \leftarrow T - \{j\}$;
 8:       search for an agent $i^* \in I \mid r_{i^*,j} \leq b_{i^*} - R_{i^*}$;
 9:       if such an $i^*$ exists then
10:         $S_j \leftarrow i^*$;
11:         $R_i \leftarrow R_i - r_{i,j}$; $R_{i^*} \leftarrow R_{i^*} + r_{i^*,j}$;
12:       end if
13:     until $T = \emptyset \lor R_i \leq b_i$;
14:   end if
15: end for
16: /* Phase 2: improve total costs */
17: for $j = 1$ to $n$ do
18:   $i \leftarrow S_j$;
19:   $i^* = \mathrm{argmin}_{i' \in I} \{c_{i',j} \mid c_{i',j} < c_{i,j} \land r_{i',j} \leq b_{i'} - R_{i'}\}$;
20:   if such an $i^*$ exists then
21:     $S_j \leftarrow i^*$;
22:     $R_i \leftarrow R_i - r_{i,j}$; $R_{i^*} \leftarrow R_{i^*} + r_{i^*,j}$;
23:   end if
24: end for
```

2.1 Genetic Algorithm from Chu and Beasley

Another hybrid GA for the GAP is due to Chu and Beasley [3, 4]. In extensive experiments, it was shown to outperform several of the previous algorithms in terms of solution quality, including in particular MTH, branch-and-bound according to [9] with a given time-limit, and the variants of simulated annealing and tabu search from [10].

This GA represents a candidate solution by a vector $S = (S_1, \ldots, S_n) \in I^J$. For each $j = 1, \ldots, n$, the value S_j indicates the agent to which job j is assigned to. This representation is compact and ensures that the assignment constraints (3) are always satisfied.

Initial candidate solutions are created uniformly at random. The GA's variation operators are standard one-point crossover and the exchange of two randomly chosen positions representing mutation. Each offspring solution is heuristically improved by Algorithm 1. In its first phase, this procedure attempts to repair infeasible solutions violating capacity constraints. This is done by trying to reassign jobs from agents with exceeded capacities to other agents. Note, however, that this approach is not always successful in making a solution feasible, since the problem of finding *any* feasible solution is in general already NP-hard. The second phase of Algorithm 1 aims on improving total costs by reassigning jobs without further violating capacity constraints. The whole algorithm runs in time $O(m^2 n)$.

To treat infeasible solutions, the GA calculates for each solution the *total capacity excess*

$$u(S) = \sum_{i=1}^{m} \max \left(0, \sum_{j \in J | S_j = i} r_{i,j} - b_i \right) \tag{5}$$

in addition to the total costs $c(S) = \sum_{j=1}^{n} c_{S_j,j}$ corresponding to equation (1). Binary tournament selection is used for selecting the parents for reproduction. In these tournaments, only the total costs are considered. In a steady state manner, a newly created and locally improved offspring immediately replaces another solution in the population. The solution to be replaced is the one with the highest capacity excess; only if all solutions in the population are feasible, costs are considered and the worst solution is replaced.

3. NEW APPROACHES

This section describes novel extensions, respectively modifications, to the GA from Chu and Beasley. They improve the average quality of final solutions and the run-time for large, hard problem instances significantly, as documented in Section 4.

3.1 Constraint-Ratio Heuristic

The *constraint-ratio heuristic* (CRH) is used to create candidate solutions for the initial population and replaces the pure random initialization. Its purpose is to provide the GA with more meaningful starting solutions. In particular, the proportion of feasible solutions is significantly increased. Nevertheless, the initial population's diversity remains relatively high. The heuristic consists of two separate strategies which are alternately applied, namely the *constraint heuristic* and the *ratio heuristic*.

The constraint heuristic represents a more intelligent random initialization which takes the capacity constraints of the agents into account, trying to find a feasible solution. The jobs are considered in random order. For each job, the procedure determines those agents to which the job may be assigned without exceeding resource capacities. From these agents, one is chosen at random, and the assignment is performed. If no such agent exists, a capacity violation cannot be avoided, and an agent is randomly picked from all agents. The creation of one solution by this heuristic takes time $O(mn)$.

The ratio heuristic considers the *relative cost-resource index* $\gamma_{i,j}$ of the assignment of job j to agent i as well as the capacity constraints of the agents. $\gamma_{i,j}$ can be interpreted as a heuristic measure for the desirability of an assignment and is calculated from the costs $c_{i,j}$ and the relative resource consumption $r^*_{i,j}$:

$$\gamma_{i,j} = c_{i,j} \cdot r^*_{i,j} \quad \text{with} \quad r^*_{i,j} = \frac{r_{i,j}}{b_i}\,. \tag{6}$$

The algorithm repeatedly chooses a yet unassigned job j at random and searches for the agent i having enough free resource capacity left for processing job j and having the smallest index γ_{i,j^*}. If no such agent exists, some other agent is randomly chosen as in the constraint heuristic. This procedure also can be implemented in time $O(mn)$.

3.2 Linear Programming Initialization

The linear programming (LP) initialization is an alternative to the constraint-ratio heuristic for creating a variety of meaningful initial solutions. The linear programming relaxation of the GAP is obtained by replacing the integer constraints (4) by $0 \le x_{i,j} \le 1$. This linear program can be solved efficiently by methods like the simplex algorithm. In general, this yields a solution x^{LP} with some fractional variable values. The following randomized rounding procedure is then applied to obtain an integral solution S.

In a first step, S adopts all integral assignments from x^{LP}: $S_j = i \leftrightarrow x^{\mathrm{LP}}_{i,j} = 1$. For each job j being fractionally assigned to several agents in x^{LP}, the agent i^* with the largest contribution is determined, and job j is completely assigned to this agent in S: $S_j = \mathrm{argmax}_{i \in I}\, x^{\mathrm{LP}}_{i,j}$.

This integral solution is typically highly infeasible due to capacity constraint violations. We apply the randomized heuristic improvement operator from Chu and Beasley (Algorithm 1), to derive a variety of more meaningful and often feasible initial solutions for the GA.

3.3 Selection and Replacement

In contrast to Chu and Beasley's cost-based parental selection and excess-based replacement scheme, we suggest a more traditional penalty-based mechanism.

Each candidate solution S gets assigned a fitness value $f(S) > 0$. If a solution is feasible, its fitness is identical to its total costs $c(S)$. For an infeasible solution, we calculate the *average relative capacity excess*

$$u_{\mathrm{R}}(S) = \frac{1}{m} \cdot \sum_{i=1}^{m} \max \left(0, \sum_{j \in J | S_j = i} \frac{r_{i,j}}{b_i} - 1 \right) \tag{7}$$

and scale it in order to ensure that this value is always larger than the fitness of any feasible solution:

$$f(S) = C_{\max} \cdot (1 + u_{\mathrm{R}}) \quad \text{with} \quad C_{\max} = \sum_{j=1}^{n} \max_{i \in I} \{c_{i,j}\}. \tag{8}$$

Based on this fitness, binary tournament selection with replacement is performed for choosing parents. A newly created offspring always replaces the worst solution in the population. To guarantee a minimum diversity, created duplicates are always immediately discarded.

Extensive preliminary experiments indicated the advantages of this selection and replacement strategies over the one proposed by Chu and Beasley. In particular the more aggressive preference of feasible solutions and the rating of infeasible solutions by means of their relative capacity excess only turned out to be crucial.

3.4 Heuristic Mutation

Instead of using standard position-wise mutation, the following more intelligent heuristic operator is proposed. A certain number n_{mut} of jobs is randomly chosen and released from the agents. These jobs are then reassigned according to Martello and Toth's heuristic (MTH), see Section 2. The main advantage of this mutation is its smaller probability to make a feasible solution infeasible or to worsen the capacity excess of an infeasible solution. Experimental results indicate that, despite the strong heuristic bias of this operator, enough random variation is introduced in order to avoid getting trapped prematurely at some poor local optimum.

4. EMPIRICAL COMPARISON

This section compares the proposed GA with its two variants for initialization to Chu and Beasley's GA and to the commercial general purpose branch-and-cut system CPLEX.

4.1 Test Problem Instances

We use the "large-size" standard test problem instances of type A to D from Chu and Beasley [4], which are electronically available at the OR-Library [2]. These instances contain up to 20 agents and 200 jobs. Since it turned out that these instances are not challenging enough for nowadays algorithms and computers, we extended the library with new instances up to 80 agents and 400 jobs. Furthermore, two new types of instances, called E and F, were derived from Chu and Beasley's most challenging type D. All new instances are electronically available at `http://www.ads.tuwien.ac.at/pub/gap`.

Type D instances were created by choosing resource requirements $r_{i,j}$ from $\{1,\ldots,100\}$ uniformly at random. Costs were correlated to the resource requirements by setting them to $c_{i,j} = 111 - r_{i,j} + U$, where U is a random integer from $\{-10,\ldots,10\}$. Resource capacities were set to $b_i = \frac{0.8}{m}\sum_{j\in J} r_{i,j}$. The new type E and F instances were derived from the type D instances by copying resource requirements and assignment costs and setting resource capacities b_i to 70% and 200% of the corresponding type D values, respectively. Therefore, instances of type E have significantly stronger capacity constraints, while type F instances are significantly looser constrained.

4.2 Setup of Algorithms

The new GA starts by creating initial solutions either via the constraint-ratio heuristic or the LP heuristic. In the following we refer to these two variants of the GA as CRH-GA and LP-GA, respectively. In any case, Chu and Beasley's heuristic improvement (Algorithm 1) is also applied to these initial solutions.

In its main loop, the GA creates one new candidate solution by always applying one-point crossover (as in Chu and Beasley's GA), the heuristic mutation operator of Section 3.4, and Chu and Beasley's local improvement. In contrast to Chu and Beasley's GA, the selection and replacement scheme described in Section 3.3 is used.

In the experiments documented in the next paragraphs, the population size was always 100. The number of jobs to be reassigned by mutation was $n_{\mathrm{mut}} = 2$. A run was terminated when 500 000 new candidate solutions had been created without improving on the so far best solution. Each GA variant performed 10 runs on each problem instance.

Chu and Beasley's GA has been re-implemented as described in [4] in order to be able to make comparisons of running times. The parameters of this GA were set according to the suggestions of Chu and Beasley.

CPLEX 8.0 from ILOG is a commercial general purpose branch-and-cut system for solving integer linear programs (ILPs). We applied it to the standard ILP-formulation (1) to (4) of the GAP without any further specific tailoring; i.e. CPLEX's standard configuration parameters were used. CPLEX was terminated after either a provable optimal solution was found, the running time exceeded three hours, or one gigabyte RAM had been exhausted. All experiments were performed on a 1.9 GHz Pentium PC.

4.3 Results

Table 1 presents average results on the 24 standard test instances of type A to D from Chu and Beasley. CPLEX was able to solve all instances of type A to C and the smallest instance of type D to provable optimality. Column IP/LP-C_{opt} denotes the corresponding integer solution values in these cases. When CPLEX was terminated due to the running time or memory limits, the value in this column denotes the optimum value of the LP-relaxation, which is a lower bound for the integer solution value.

The quality of a solution is indicated in the table as percentage gap between the solutions's total costs $c(S)$ and either the optimum solution value (if available) or the LP-bound: $\%\text{-}gap(S) = (c(S) - C_{\mathrm{opt}})/C_{\mathrm{opt}} \cdot 100\%$. The table shows for CPLEX these percentage gaps of the best solutions obtained when it had been aborted. For Chu and Beasley's GA and CRH-GA, the table contains for each problem instance the average percentage gap of the runs' final solutions, the corresponding standard deviation (*stddev*), the percentage gap of the best run's solution ($\%\text{-}gap_{\mathrm{best}}$), and the average number of iterations and average CPU-time until the final solutions had been identified (*iter*, respectively t).

Both GAs could find optimum solutions for all instances of type A. For the other test cases, it can be observed that the average qualities of final solutions of CRH-GA are almost always better than those of Chu and Beasley's GA. The last column of the table ($\%\text{-}p$) lists error probabilities obtained from statistical t-tests for the assumption that there exist differences in the average percentage gaps of the two GAs. For more than the half of the instances of type B to D, these error probabilities are smaller than 1%. Furthermore, it can be observed that CRH-GA is often significantly faster than Chu and Beasley's GA. CPLEX, however, is a strong competitor. Even for the instances where it could not find the optimum, its final solutions were better than those of both GAs with only a single exception.

In Table 2, CPLEX is compared to CRH-GA and LP-GA on the newly created larger instances of type D, E, and F. For the two GAs and each problem instance, average values are listed again for the percentage gaps of the runs best solutions and the number of iterations and CPU-times to find them. Now, CPLEX was able to solve 9 smaller instances to provable optimality. For larger instances, CRH-GA yielded occasionally better results than CPLEX. LP-GA turned out to be superior in most cases. For 27 out of all 45 instances, LP-GA returned on average the best results. The last column $\%\text{-}p$ shows error probabilities of t-tests for the assumption that there exist differences in the average percentage gaps of CRH-GA and LP-GA. The generally small values strongly indicate LP-GA's superiority. Furthermore, LP-GA was almost always significantly faster than CRH-GA. The reason is the LP initialization's stronger ability to create highly fit, feasible initial solutions. On average over the six type D instances of Chu and Beasley, about 99% of the solutions created by the LP initialization were feasible, while for the CRH initialization, this factor is only $\approx 90\%$. The average percentage gap of the solutions created by the LP initialization was 1.6%, while it was 12.1% when using the CRH initialization. This substantially higher solution quality of the LP initialization obviously also compensates the disadvantage of the smaller diversity in the initial population.

Table 1: Computational results of CPLEX, Chu and Beasley's GA, and CRH-GA for Chu and Beasley's instances of type A to D; ○: integer optimum reached, i.e. %-$gap = 0$.

Prob.	Size		IP/LP	CPLEX	Chu and Beasley's GA					CRH-GA					
Type	m	n	C_{opt}	%-*gap*	%-*gap*	*stddev*	%-gap_{best}	*iter*	t [s]	%-*gap*	*stddev*	%-gap_{best}	*iter*	t [s]	%-p
A	5	100	1 698	○	○	0.00	○	1 469	1.9	○	0.00	○	524	1.0	50.00
		200	3 235	○	○	0.00	○	173	7.0	○	0.00	○	181	25.9	50.00
	10	100	1 360	○	○	0.00	○	356	7.3	○	0.00	○	9 046	97.1	50.00
		200	2 623	○	○	0.00	○	1 447	28.5	○	0.00	○	2 531	10.2	50.00
	20	100	1 158	○	○	0.00	○	404	28.5	○	0.00	○	688	22.2	50.00
		200	2 339	○	○	0.00	○	444	112.7	○	0.00	○	712	24.2	50.00
B	5	100	1 843	○	0.35	0.31	○	309 088	59.3	0.03	0.08	○	210 424	51.2	0.61
		200	3 552	○	0.30	0.15	0.03	780 172	240.9	0.12	0.05	0.03	362 588	165.4	0.22
	10	100	1 407	○	0.07	0.07	○	76 341	20.9	○	0.00	○	42 048	30.9	0.75
		200	2 827	○	0.34	0.13	0.14	807 095	335.0	0.26	0.14	0.07	509 780	381.8	10.20
	20	100	1 166	○	0.07	0.06	○	101 031	58.2	0.14	0.21	○	120 872	189.3	18.89
		200	2 339	○	0.10	0.05	0.04	167 451	204.0	0.12	0.08	0.04	212 798	378.5	25.50
C	5	100	1 931	○	0.38	0.21	○	612 830	104.1	0.15	0.18	○	165 421	39.2	1.25
		200	3 456	○	0.23	0.11	0.06	1 063 783	335.8	0.07	0.03	0.03	707 290	320.0	0.06
	10	100	1 402	○	0.29	0.26	0.07	556 707	123.4	0.11	0.09	0.07	137 447	54.1	4.14
		200	2 806	○	0.48	0.13	0.29	1 203 979	492.4	0.22	0.16	0.04	408 855	304.6	0.06
	20	100	1 243	○	0.51	0.26	0.08	363 620	142.7	0.26	0.13	○	342 068	289.6	0.99
		200	2 391	○	0.62	0.19	0.25	930 809	649.6	0.56	0.18	0.21	827 683	1 140.1	23.79
D	5	100	6 353	○	0.66	0.21	0.31	989 664	195.3	0.31	0.06	0.19	1 248 014	327.4	0.03
		200	12 736.2	0.09	0.71	0.15	0.47	1 894 286	676.4	0.39	0.16	0.24	1 635 381	814.7	0.02
	10	100	6 323.4	0.72	1.65	0.38	0.88	2 016 763	591.4	1.24	0.22	0.78	1 083 907	472.6	0.74
		200	12 418.3	0.30	1.68	0.15	1.47	3 816 928	2 085.8	1.17	0.18	0.95	2 259 536	1 909.7	< 0.01
	20	100	6 142.5	2.37	2.70	0.37	2.06	2 491 521	1 272.0	2.36	0.31	1.90	1 126 251	902.3	2.41
		200	12 217.7	1.03	2.54	0.30	1.92	5 294 138	5 039.3	2.00	0.33	1.38	2 457 659	3 825.7	0.09

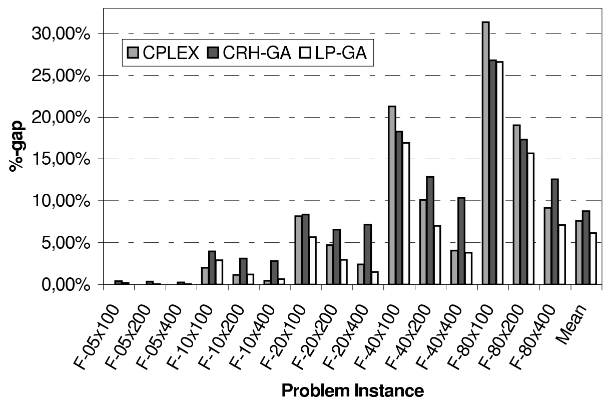

Figure 1: Average qualities of final solutions from CPLEX, CRH-GA, and LP-GA for type F instances.

Instances of type F, which have the loosest capacity constraints, seem to be hardest because the gaps of final solutions are generally significantly larger than those of type D or type E instances. Figure 1 shows the (average) qualities of final solutions from CPLEX, CRH-GA, and LP-GA for type E instances graphically.

More detailed results of further experimental investigations can be found in [5]. In particular, Chu and Beasley's GA performed consistently worse than CRH-GA and LP-GA also on the new instances, and LP-GA yielded consistently better results than both other GAs also on Chu and Beasley's smaller instances. Furthermore, it could be proven that each of the suggested new approaches of Section 3, i.e. the new initialization schemes, the new selection and replacement scheme, and heuristic mutation, lead to an improvement of Chu and Beasley's original GA independently of each other.

5. CONCLUSIONS

This article proposes several improvements for a previously published, successful genetic algorithm from Chu and Beasley for the generalized assignment problem. Firstly, the suggested constraint-ratio and linear programming heuristics create more promising initial solutions which are mostly feasible. Starting with these solutions usually speeds up the genetic algorithm and leads to significantly better final solutions. In particular on larger and more difficult problem instances, the LP heuristic turned out to be superior to the constraint-ratio heuristic, despite it results in a smaller diversity of the initial population.

The second proposed improvement lies in the modified selection and replacement strategy, which assigns infeasible solutions a fitness value depending only on the relative capacity excess and always ranks them worse than any feasible solution. In this way, infeasible solutions are more aggressively eliminated. Finally, a new effective mutation operator based on the heuristic of Martello and Toth has been proposed.

Empirical results indicate that today's general purpose branch-and-cut systems are able to solve relatively large instances of the general assignment problem to provable optimality. For hard problem instances in which resource requirements are correlated to costs, the number of agents exceeds 5, and the number of jobs exceeds 100, the used branch-and-cut solver CPLEX was, however, not able anymore to identify global optimal solutions. In these cases, the proposed genetic algorithm, in particular the variant using LP initialization, yields significantly better results in typically shorter time.

6. REFERENCES

[1] M. Amini and M. Racer. A rigorous computational comparison of alternative solution methods for the

Table 2: Computational results of CPLEX, CRH-GA, and LP-GA for new instances; ∘: integer optimum reached, i.e. %-$gap = 0$.

Prob.	Size		IP/LP	CPLEX	CRH-GA			LP-GA			
Type	m	n	C_{opt}	%-gap	%-gap	$iter$	t [s]	%-gap	$iter$	t [s]	%-p
D	5	100	6 353	∘	0.31	1 248 014	327.4	0.08	308 338	82.9	< 0.01
		200	12 736.2	0.09	0.39	1 635 381	814.7	0.15	424 045	223.1	0.05
		400	25 670	∘	0.44	3 495 381	3 609.7	0.03	241 022	268.2	< 0.01
	10	100	6 323.5	0.72	1.24	1 083 907	472.6	0.75	299 368	132.2	< 0.01
		200	12 418.4	0.30	1.17	2 259 536	1 909.7	0.26	588 684	511.9	< 0.01
		400	25 274.8	0.18	1.31	4 061 815	7 042.7	0.16	633 166	1 139.7	< 0.01
	20	100	6 142.5	2.37	2.36	1 126 251	902.3	1.67	656 233	515.2	< 0.01
		200	12 217.7	1.03	2.00	2 457 659	3 825.7	0.81	633 267	1 038.3	< 0.01
		400	24 546.8	0.51	1.97	4 041 905	13 049.6	0.44	876 090	2 988.9	< 0.01
	40	100	6 092.0	3.96	3.72	1 361 463	2 072.9	3.37	784 784	1 206.7	0.48
		200	12 244.9	2.22	3.06	2 357 294	7 116.2	1.63	902 377	3 034.4	< 0.01
		400	24 371.8	1.10	2.81	3 574 368	23 374.1	0.86	1 325 198	9 049.5	< 0.01
	80	100	6 110.5	6.60	7.01	846 320	2 878.6	7.01	948 131	3 047.5	49.15
		200	12 132.3	2.87	3.76	2 001 149	13 075.6	3.05	1 377 981	9 664.6	0.01
		400	24 177.0	2.00	3.02	3 195 382	55 945.1	1.57	2 370 939	41 258.3	< 0.01
E	5	100	7 757	∘	0.24	552 212	147.1	0.07	232 404	65.3	< 0.01
		200	15 611	∘	0.23	1 281 834	669.8	0.04	211 499	120.6	< 0.01
		400	30 794	∘	0.28	2 744 401	3 026.7	0.03	192 115	231.9	< 0.01
	10	100	7 387.8	0.61	0.91	1 005 960	447.3	0.60	359 258	170.3	0.04
		200	15 039.8	0.25	0.96	1 807 355	1 557.2	0.24	437 348	408.9	< 0.01
		400	29 977.9	0.09	0.94	3 390 020	6 136.3	0.11	591 807	1 126.7	< 0.01
	20	100	7 348.2	1.32	1.74	1 458 153	1 171.4	0.97	891 059	746.0	< 0.01
		200	14 765.2	0.89	1.70	1 955 937	3 119.4	0.61	1 041 321	1 742.7	< 0.01
		400	29 500.3	0.34	1.60	4 255 476	14 462.3	0.33	787 977	2 892.4	< 0.01
	40	100	7 316.1	3.32	3.11	976 572	1 573.5	2.86	976 760	1 600.8	3.18
		200	14 630.4	1.85	2.20	1 889 586	6 061.5	1.32	1 164 883	3 899.2	< 0.01
		400	29 186.6	0.69	2.14	2 997 713	20 536.8	0.63	1 099 475	8 387.9	< 0.01
	80	100	7 650	∘	0.78	737 647	2 614.1	0.57	806 626	2 862.6	0.04
		200	14 566.7	2.17	2.93	1 584 730	11 078.3	2.43	1 360 348	9 879.0	0.03
		400	29 161.3	1.57	2.49	3 063 944	56 250.8	1.20	2 063 900	34 618.8	< 0.01
F	5	100	2 755	∘	0.41	507 714	123.0	0.18	90 928	21.6	2.36
		200	5 294	∘	0.35	647 177	293.2	0.06	307 978	137.0	0.03
		400	10 745	∘	0.25	1 235 605	1 130.0	0.05	199 224	196.7	< 0.01
	10	100	2 276.8	1.99	3.95	1 019 383	386.4	2.88	365 538	135.9	< 0.01
		200	4 644.6	1.13	3.12	1 565 513	1 127.4	1.22	702 420	505.4	< 0.01
		400	9 372.7	0.46	2.78	2 897 607	4 207.9	0.65	1 154 144	1 663.1	< 0.01
	20	100	2 145.1	8.15	8.38	1 514 783	1 010.3	5.64	536 732	362.1	< 0.01
		200	4 310.1	4.73	6.55	2 227 943	2 878.4	2.96	760 128	1 021.5	< 0.01
		400	8 479.4	2.38	7.15	3 984 825	10 390.0	1.51	1 090 608	3 034.6	< 0.01
	40	100	2 110.1	21.28	18.27	1 191 719	1 572.9	16.94	1 090 204	1 395.7	1.64
		200	4 086.5	10.14	12.86	2 461 344	6 228.2	7.02	788 030	2 213.0	< 0.01
		400	8 274.3	4.05	10.36	3 676 777	19 192.9	3.80	1 202 311	7 212.2	< 0.01
	80	100	2 064.4	31.37	26.77	1 620 919	3 640.1	26.57	1 092 314	2 932.0	32.82
		200	4 123.4	19.05	17.33	1 738 801	9 828.6	15.67	2 624 327	13 613.2	0.32
		400	8 167.1	9.18	12.55	2 661 273	32 432.1	7.12	1 457 916	22 542.5	< 0.01

generalized assignment problem. *Management Science*, 40:868–890, 1994.

[2] J. E. Beasley. OR-Library. Imperial College of Science, Technology and Medicine, London, U.K. `http://mscmga.ms.ic.ac.uk/info.html`.

[3] P. C. H. Chu. *A Genetic Algorithm Approach for Combinatorial Optimisation Problems.* PhD thesis, Imperial College of Science, Technology and Medicine, London, U.K., 1997.

[4] P. C. H. Chu and J. E. Beasley. A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, 24:17–23, 1997.

[5] H. Feltl. *Ein Genetischer Algorithmus für das Generalized Assignment Problem.* PhD thesis, Vienna University of Technology, Vienna, Austria, 2003.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W.H. Freeman, San Francisco, 1979.

[7] K. Jörnsten and M. Näsberg. A new Lagrangian relaxation approach to the knapsack problem. *European Journal of Operational Research*, 27:313–323, 1986.

[8] S. Martello and P. Toth. Algorithm for the solution of the 0–1 single knapsack problem. *Computing*, 21:81–86, 1978.

[9] S. Martello and P. Toth. An algorithm for the generalized assignment problem. *Operational Research*, 81:589–603, 1981.

[10] I. H. Osman. Heuristics for the generalised assignment problem: Simulated annealing and tabu search approaches. *OR Spektrum*, 17:211–225, 1995.

[11] M. W. P. Savelsbergh. A branch-and-cut algorithm for the generalized assignment problem. *Operations Research*, 45/6:831–841, 1997.