Ninjas' Revenge: the secret genetic technique

Cristian Andres Camargo Giraldo CRISTIAN.ANDRES.CAMARGO@ESTUDIANTAT.UPC.EDU

Rodrigo Pablo Carranza Astrada RODRIGO.PABLO.CARRANZA@ESTUDIANTAT.UPC.EDU

Santiago del Rey Juárez SANTIAGO.DEL.REY@ESTUDIANTAT.UPC.EDU

Yazmina Zurita Martel YAZMINA.ZURITA@ESTUDIANTAT.UPC.EDU

Abstract

This is a great project and therefore it has a concise abstract.

Keywords: List of keywords

1. Problem statement and goals

One of the most popular problems studied in constrained combinatorial optimization is the knapsack problem. Given a finite set of objects with associated weights and values, the objective is to maximize the value of a collection formed by these items without exceeding a predefined weight limit.

The knapsack problem has diverse practical applications which makes it particularly interesting. For example, it has been applied to production and inventory management Ziegler (1982), financial models Mathur et al. (1983) and queueing operations in computer systems Gerla and Kleinrock (1977) or manufacturing Bitran and Tirupati (1989).

There are several variations to the knapsack problem. We could consider there are multiple copies of each item or take into consideration their volume in addition to their weight. However, we will focus on the simplest case, the one dimensional 0-1 knapsack problem, where the only constraint is the weight and the number of copies of each item is limited to 1.

Although the premise might look simple on the surface, this is actually an NP-hard problem. Thus, there is no known algorithm that achieves optimal solutions in polynomial time for all cases of this problem. Still, a sufficiently good solution can be found quickly by resorting to heuristics methods. The one that we will explore is an effective and commonly used metaheuristic known as genetic algorithm.

Genetic algorithms (GAs) belong to the family of techniques known as evolutionary algorithms. They draw inspiration from natural selection and genetic processes to provide solutions to complex optimization problems and model evolutionary systems.

The main process can be described as an evolutionary cycle. They first initialise a population of chromosomes, which represent candidate solutions. Some of these individuals will be drawn from the population by means of a selection mechanism and compared according to their fitness. In the reproduction phase, the genetic material of the best individuals will be combined to form offspring. Also, some values of the offspring's chromosomes will be mutated. Finally, a replacement strategy is set to generate the next generation population. This process is repeated for a number of generations or until some convergence criterion is met.

A solution to the knapsack problem can be represented as a sequence of binary values $\{p_1, p_2, ..., p_m\}$, where m is the number of objects. A value $p_i = 1$ would mean that the solution contains the object p_i and $p_i = 0$ that it does not. We will apply GAs to eighteen different cases, each of them specified by a file containing the number of objects m and the capacity K of a knapsack in the first line and the value v_i and weight w_i of each object p_i in subsequent lines.

The goal of this project is to find the optimal solution to each of the eighteen knapsack problems proposed by applying GAs. To this end, we will have to choose and implement suitable mechanisms in each of the phases of the evolutionary cycle. The description of these methods, as well as the reasons for their election, are detailed in Section 3.

2. Previous work

This is a very important part, because it puts your work in context.

3. The CI methods

When applying genetic algorithms to a problem, there are several aspects to consider, each one with its own methods. In this section, we define the different methods used in our implementation for each one of these aspects.

3.1. Population initialization

For the initialization of the population, we have used a random initialization approach, which has been traditionally used in GAs for its simplicity and efficiency. However, we introduced an initialization range parameter in order to give the algorithm better chances of finding the optimal solution in case we know where this solution should be. This initialization range in conjunction with the possibility of sorting the input items by their value/weight rate can highly improve the algorithm performance.

3.2. Fitness function and selection

To be able to assess how well an individual (i.e. possible solution) fits our objective we must first define the fitness function. For our particular problem, we defined the following fitness function:

$$F(x) = \begin{cases} -\infty & if\, x \bullet w > c \\ x \bullet v & otherwise \end{cases}$$

, where x is the bitstring representing the individual, w is the list of item weights, v is the list of item values and c is the knapsack capacity.

Once we have the fitness function we are able to proceed to the selection step. We decided to implement the tournament and elitism selection methods for our algorithm. Particularly, we use a fixed value of $k = 2$ for both tournament and elitism. We decided to use tournament selection instead of other methods such as standard roulette wheel or rank selection because of its simplicity and because seems to outperform the other approaches Razali et al. (2011). On the other hand, elitism was implemented because of its usage in the tournament selection method.

3.3. Crossover and mutation

Several crossover techniques can be applied in GAs such as one-point, 2-point, multi-point (with more than two cut points) and uniform crossovers. In this work, we have implemented the one-point, 2-point and uniform crossovers. The reason behind this choice is the simplicity of these techniques and their good performance, especially the 2-point crossover Hasançebi and Erbatur (2000); DeJong and Spears (1990); Adeli and Cheng (1993).

For the mutation step, we implemented a simple bit-flip where a random number of genes between 1 and the size of the chromosome are flipped. In addition, we set the mutation probability to 0.5 since lower values made the algorithm get stuck in local minimums more frequently.

3.4. Replacement strategy

The final step of the GA is the replacement strategy used to obtain the new generation. In our case, we used elitism to replace the worst individuals of the current population. With this strategy, in some cases could happen that the old generation is entirely replaced. In addition to the elitism, we also decided to remove repeated individuals before applying the replacement. This decision was made to ensure that the diversity of the population is maintained. Additionally, we defined an extra case that does not use elitism. This is when there is no good solution (i.e. the fitness value is $-\infty$) neither in the parents nor the offspring. In this case, we decided to obtain the new generation by selecting, randomly, half of the individuals from μ and the other half from λ.

4. Results and Discussion

The main part of the document.

5. Strengths and weaknesses

Be critic with your work ...

6. Conclusions and future work

The conclusions are not a mere repetition of the abstract. Basically, you should describe “what you know now that you did *not* before doing the work”. In addition, mention what would be natural follow-up lines of work.

References

References

Hojjat Adeli and Nai-Tsang Cheng. Integrated genetic algorithm for optimization of space structures. *Journal of Aerospace Engineering*, 6(4):315–328, 1993.

Gabriel R Bitran and Devanath Tirupati. Tradeoff curves, targeting and balancing in manufacturing queueing networks. *Operations Research*, 37(4):547–564, 1989.

K DeJong and WM Spears. An analysis of the interacting roles of the population size and crossover type in genetic algorithms. *Parallel problem solving from nature*, pages 38–47, 1990.

Mario Gerla and Leonard Kleinrock. On the topological design of distributed computer networks. *IEEE Transactions on communications*, 25(1):48–60, 1977.

Oğuzhan Hasançebi and Fuat Erbatur. Evaluation of crossover techniques in genetic algorithm based optimum structural design. *Computers & Structures*, 78:435–448, 2000. ISSN 0045-7949. doi: https://doi.org/10.1016/S0045-7949(00)00089-4. URL https://www.sciencedirect.com/science/article/pii/S0045794900000894.

Kamlesh Mathur, Harvey M Salkin, and Susumu Morito. A branch and search algorithm for a class of nonlinear knapsack problems. *Operations Research Letters*, 2(4):155–160, 1983.

Noraini Mohd Razali, John Geraghty, et al. Genetic algorithm performance with different selection strategies in solving tsp. In *Proceedings of the world congress on engineering*, volume 2, pages 1–6. International Association of Engineers Hong Kong, 2011.

Hans Ziegler. Solving certain singly constrained convex optimization problems in production planning. *Operations Research Letters*, 1(6):246–252, 1982.

Appendix A. Proof of theoretical results

(if applicable)

Appendix B. Implementation details

(if applicable)