A Particle Swarm Optimization Algorithm Implemented on MultiAgent Systems Applied to a Sugar Cane Distribution Problem

Nicolás Majorel Padilla[1,3]
nicolas.majorel@gitia.org

Pedro Araujo[1]
pedro.araujo@gitia.org

Adrián Will[1,3]
adrian.will@gitia.org

Sebastián Rodríguez[1,2]
sebastian.rodriguez@gitia.org

[1]Grupo de Investigación en Tecnologías Avanzadas de Tucumán (GITIA)
Universidad Tecnológica Nacional - Facultad Regional Tucumán
Rivadavia 1050 - S.M. de Tucumán - Tucumán - Argentina

[2]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Av. Rivadavia 1917 - Ciudad Autónoma de Buenos Aires - Argentina

[3]Universidad Nacional de Tucumán - Facultad de Ciencias Exactas y Tecnología
Av. Independencia 1800 - S.M. de Tucumán - Tucumán - Argentina

Abstract

Sugar cane distribution optimization problem is a critical problem in Tucumán, Argentina, with geographical, environmental and economic characteristics that makes it different from similar transportation or supply chain problems.

Motivated by this problem, we introduce in this paper a new heuristic algorithm to solve it. The problem is represented as a non-linear variant of the Generalized Assignment Problem, and the algorithm introduced is a Binary Particle Swarm Optimization implemented on a MultiAgent System, written in SARL language. The new algorithm was tested on a previously developed benchmark suite, and it can efficiently solve the problem, finding the optimal solution for all test instances of different types.

1 Introduction

In a previous work presented by the authors [Majorel Padilla et al., 2015], a non-linear variant of the well known *Generalized Assignment Problem (GAP)* was introduced, as well as a benchmark suite with its optimal results, for using as a reference for future algorithms. Moreover, a model of the problem was presented, using the organizational approach from MultiAgent Systems (MAS).

In: Yu. G. Evtushenko, M. Yu. Khachay, O. V. Khamisov, Yu. A. Kochetov, V.U. Malkova, M.A. Posypkin (eds.): Proceedings of the OPTIMA-2017 Conference, Petrovac, Montenegro, 02-Oct-2017, published at http://ceur-ws.org

This variant was designed to solve the sugar cane distribution optimization problem in Tucumán, a province in the northwest of Argentina. Being one of the largest industries in the province, the distribution of sugar canes to sugar mills lacks an optimized scheduling of the complete sugar cane supply chain. This fact not only reduces profits to both sugar cane farmers and sugar mill owners, but also has an important impact on highway traffic, road safety and environmental pollution.

There have been several efforts to optimize similar sugar cane supply chain optimization problems throughout the world [Giles et al., 2009, Le Gal et al., 2009, Yosnual & Supsomboon, 2004]. However, because of the substantial differences between the supply chain processes among countries, the solutions proposed in those cases can not be applied directly.

Considering this background, this paper proposes a new heuristic algorithm for the aforementioned problem. The resulting algorithm is a Particle Swarm Optimization (PSO), in its Binary version, implemented on a MultiAgent System. The benchmark suite developed for the problem is used to test the new algorithm, and the results of these tests are exposed.

The new algorithm was written using SARL [Rodriguez et al., 2014]. SARL is the first general purpose language for agent oriented programming, based on the Java language, fully independent of any other platform or any other agent architecture. Through its Capacity-Role-Interaction-Organization metamodel, SARL provides a well defined set of basic concepts (agent oriented first-class abstraction) needed to implement MAS. It is an effort to improve user experience in developing complex systems, thanks to its simple and extensible syntax.

The rest of the paper is structured as follows: in Section 2 a brief recapitulation of the GAP is presented, as well as the specific variant which constitutes the problem to solve. This section also describes some related works between MAS and PSO. Next, Section 3 explains the basics of PSO and details about the MAS implementation are introduced. Finally, in Section 4 the testing methodology is described and the results presented, with final conclusions and future work in Section 5.

2 Related Work

In this section we will give a brief overview of the GAP, and the variant introduced to solve the proposed problem, as well as related works of MAS and PSO which were used as basis for the proposed algorithm.

2.1 GAP: Definition and Variants

The Generalized Assignment Problem is defined in [Chu & Beasley, 1997] as a combinatorial optimization problem, which consists in finding the assignment of tasks into agents that minimizes cost, with the constraint that each task is assigned to exactly one agent, and subject to agent's capacity. For the rest of this paper, we follow the notation defined for GAP in [Ozbakir et al., 2010]: n represents the number of agents, m is the number of tasks; when task i is assigned to agent j, this assignation is expressed as x_{ij}, and it has an associated cost c_{ij} or profit p_{ij}. Besides, each agent has a maximum capacity b_j, and task i consumes a_{ij} of agent j resources.

GAP is an NP-hard problem [Fisher et al., 1986], and finding if a valid solution exists is NP-complete [Martello & Toth, 1990]. Many deterministic and heuristic algorithms can be found in literature [Cattrysse & Van Wassenhove, 1992, Osman, 1995, Ramalhinho & Serra, 2008, Yagiura et al., 1998], and also many variants [Krumke & Thielen, 2013, Laguna et al., 1995, C. Rainwater & Romeijn, 2009].

Nevertheless, none of these variants matches all the features required by the particular problem we were trying to solve, so we proposed a new variant that models sugar mills as GAP agents and sugar cane farms as GAP tasks, and called this variant Variable Profit GAP (VPGAP). In VPGAP, the assignment x_{ij} represents when a sugar cane farm i is assigned to sugar mill j, it has an associated profit p_{ij}, and this sugar cane farm provides a_{ij} tons of sugar cane to the mill for processing. The sugar mill's maximum capacity is modeled as b_j, and it also has the following distinct features:

- Each agent (sugar mill) has also a minimum capacity, d_j.
- The agents (sugar mills) have a non-linear efficiency, h_j, related to the amount of resources (sugar cane) consumed by the tasks (sugar cane farms) associated to it.
- Agent's peak efficiency is about 85% of its maximum capacity. Its efficiency is constantly increasing until this value, and constantly decreasing after it.
- The variant is limited to model only one day at the time.

The VPGAP variant introduced in [Majorel Padilla et al., 2015] is expressed as:

$$\text{maximize } Z(X) = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}x_{ij}h_j \left(\sum_{i=1}^{n} a_{ij}x_{ij}\right) \tag{1}$$

subject to:

$$d_j \leq \sum_{i=1}^{n} a_{ij}x_{ij} \leq b_j \quad j = 1, \ldots, m \tag{2}$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad i = 1, \ldots, n \tag{3}$$

$$x_{ij} = \begin{cases} 1, & \text{if task i is assigned to agent j} \\ 0, & \text{in other case.} \end{cases} \tag{4}$$

where $Z(X)$ is the system's global efficiency, and h_j represents agent's own efficiency, as a percentage of its maximum capacity. The function used in this work is a Gaussian with a mean of 0.85 and a standard deviation of 0.25, in the interval [0.30, 1.00] and 0 outside this interval. This simple function accomplishes with the constraints from above.

2.2 MultiAgent Systems and PSO

Over the past three decades, MultiAgent Systems emerged as a new computational paradigm, and have been used in a wide range of complex problems, from robotics to distributed problem solving [Wooldridge, 2009]. The paradigm's main concept is the *agent*, which is defined as "a physical or virtual entity with a high degree of autonomy, independent, capable of cooperating, competing, communicating, acting and taking control of its own behavior within its own goals" [Weiss, 2013]. These entities operate within an environment, which they are able to perceive and to modify. Besides, agents can group themselves into societies, in which they can cooperate with each other to reach common or individual goals which could not be achieved individually. Therefore, it is commonly expressed that the solution emerges as result of the interaction among agents.

However, only a few examples can be found in the literature of MAS in conjunction with PSO. One of these examples is [Zhao et al., 2005], whose MultiAgent based Particle Swarm Optimization (MAPSO) was used to find optimal solution to the distribution of reactive energy in power systems, with excellent results. A different example is found in [Pugh & Martinoli, 2007], where a multi-robot search algorithm is used to find optima in a multi- dimensional function space, inspired by PSO, and with several advantages over similar methods. In [Wang et al., 2010] a MAS is used for remote house control, while PSO is used to find the optimal set of control points of the system. Finally, a new algorithm named HMAPSO (Hybrid MultiAgent-based Particle Swarm Optimization) was introduced in [Kumar et al., 2011] to find a solution for the economic problem of energy distribution.

3 Proposed Algorithm

In this section, the proposed algorithm is presented. First we describe the general PSO algorithm, its components and behaviors, and then Binary PSO algorithm is described. Finally, the mixing between Binary PSO and MAS is introduced.

3.1 PSO Overview

Particle Swarm Optimization algorithm [Kennedy & Eberhart, 1995] is a bio- inspired algorithm, based on a meta-heuristic inspired in social behavior of certain species, such as flock of birds' flight, or fish bank movement, and it has grown in popularity thanks to its quick convergence and its simple implementation. In the literature, PSO is used in traditional optimization [Hu et al., 2004], in multi-objective optimization [Zhang et al., 2003], and in dynamic optimization [Blackwell, 2007], among others.

PSO is a population based algorithm, but it is not an evolutionary algorithm, because it lacks a selection mechanism. In PSO, every particle (or agent) represents a possible solution to the problem, it is part of a swarm

of particles with similar characteristics, and it has to make decisions regarding its behavior. These decisions are made taking into account two major components: an individual component of each particle, representing results obtained in past explorations, and a social component, representing the influence of the rest of the particles of the swarm. With these two components, the particle can determine where it will move to reach a new solution of the problem solutions space.

It is important to note that in PSO the population is fixed-sized, the solution's space is predefined (the particles are not allowed to move outside this boundary), and the particles are constantly searching for new solutions. Moreover, PSO makes little assumptions about the problem to be solved, and is capable of searching in very large solution's spaces. As a disadvantage, PSO can not guarantee that the optimal solution to the problem can be found.

In its more general form, each PSO particle is composed by the following elements: a vector X_i representing the actual position of the particle; a vector $pBest_i$ holding the particle's best solution found at the moment; a velocity vector V_i holding the particle's future direction; a $fitness\_x_i$ value, indicating how good the actual solution is; and a $fitness\_pBest_i$ value, indicating how good the particle's best solution found at the moment is.

The population is initialized with random positions and velocities for each particle of the swarm. Then, in an iterative process, particles start to move in the search space until a new position is found. In this new position, the value $fitness\_x_i$ is calculated, and if this value is better than the best fitness found until then, $pBest_i$ and $fitness\_pBest_i$ are updated.

In each iteration, the particle's velocity is updated following equation 5, which is comprised of three terms: the first one represents *inertia*, the particle's tendency to keep moving in the same direction [Shi & Eberhart, 1998]; second one represents *history*, the particle's tendency to stay closer to positions that produced good results previously; and the third one represents the influence of the rest of the swarm, or the imitation of the best particle of the swarm (variable $gBest_i$ is the value of the best solution found at the moment by the rest of the swarm particles).

$$\begin{aligned} V_i(t+1) = & \; w * V_i(t) \\ & + c_1 * rand_1 * (pBest_i - X_i(t)) \\ & + c_2 * rand_2 * (gBest_i - X_i(t)) \end{aligned} \tag{5}$$

As the vectors change in every iteration, the variable t is used to indicate time. The value w is named *inertial factor*. c_1 and c_2 are constants known as *speedup constants* and they usually have the same value. $rand_1$ and $rand_2$ are random numbers between 0 and 1.

When every particle has updated its velocity, it sums up with the actual position in order to determine the new position for the next iteration, as stated in equation 6.

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{6}$$

Many articles can be found in the literature related to the correct choice of w, c_1 and c_2 values, which are critical to the algorithm's convergence [Clerc & Kennedy, 2002, Van Den Bergh, 2002]. In [Shi & Eberhart, 1998] the authors showed that when w is small, the algorithm makes a more localized search and it is strongly dependent of the initialization. While w increases, the search widens, but it is more difficult to find the optimal value. So, it is recommended that the value of w decreases over time. But later, in [Shi & Eberhart, 1999] and in [Zhang et al., 2003], authors concluded that whether the value of w should increase or decrease over time depends on the problem to be solved.

3.2 Binary PSO

In [Kennedy & Eberhart, 1997], Kennedy proposed a binary version of the PSO algorithm, with the goal of applying PSO to combinatorial optimization. This modification affects each particle's codification, and the position updating equation. In Binary PSO, the velocity is used as an input to a sigmoid function indicating the probability that the position takes the value 1. Then, a random number is generated and the new position is determined by equation 7.

$$X_i(t+1) \;\; = \;\; \begin{cases} 1, & \text{if } rand() < sig(V_i(t+1)) \\ 0, & \text{other.} \end{cases} \tag{7}$$

A Binary PSO algorithm pseudo-code is shown in Algorithm 1.

Data: Pop
Result: best solution found
Pop = CreatePopulation(N);
while *termination condition not reached* **do**
  for *i = 1 to size(Pop)* **do**
    Evaluate particle X_i of Pop;
    if *fitness(X_i) is better than fitness($pBest_i$)* **then**
      $pBest_i = X_i$;
      fitness($pBest_i$) = fitness(x_i);
    end
  end
  for *i = 1 to size(Pop)* **do**
    Choose $gBest_i$;
    Calculate new V_i;
    for *d = 1 to n* **do**
      if *rand() < sig(V_{id})* **then**
        $X_{id} = 1$
      end
      else
        $X_{id} = 0$
      end
    end
  end
end
Algorithm 1: Binary PSO algorithm pseudo-code.

3.3 MAS Link-up

With the approach explained in previous sections, there exists a one-to-one relationship between a PSO particle and an MAS agent. At the same time, each one of these agents is identified as a possible solution for the optimization problem. In our proposed MAS model, agents should have one of two different roles: *Swarm* or *Particle*. In Figure 1 a simple scheme shows the interaction between the two agents and their methods and required capacities.

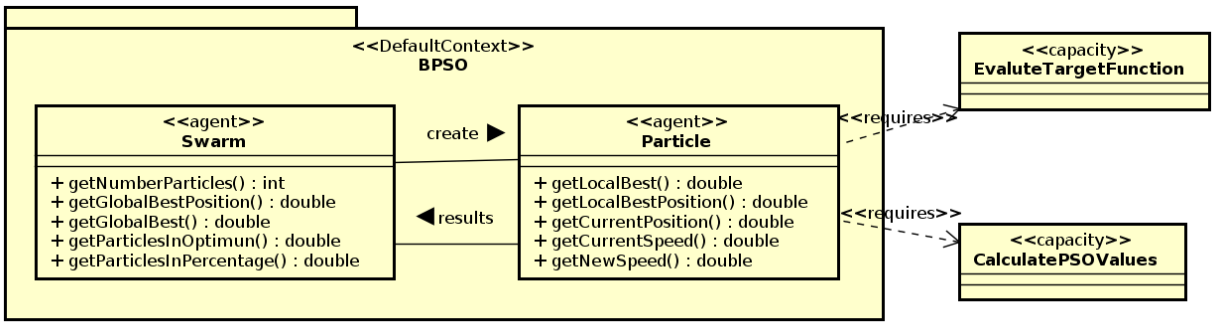

Figure 1: Agent's class diagram.

The *Swarm* agent has a unique instance in simulation time, and is responsible for the creation of all of the instances of *Particle* agents defined in the population. Besides, *Swarm* agent receives from every *Particle* agent the actual position values, X_i.

Each *Particle* agent receives from the *Swarm* agent its corresponding initialization values, and with these values it calculates its new velocity V_i, its new position X_i and its new $fitness\_x_i$ value. This $fitness\_x_i$ value

is also passed to the *Swarm* agent so it can determine which of the *Particle* agents has reached the best solution so far.

Thanks to the interoperability of SARL with Java (the possibility to create Java objects from SARL, and invoke its methods from SARL-defined agents in a completely transparent way), the code migration from [Majorel Padilla et al., 2015], written for Janus platform, was simple and straightforward. Additionally, SARL is fully distributed and allows for parallel execution of agents' behavior and automatic detection of execution kernels, features that can be used for distributed computation of simulation.

The code implemented is open-source and can be found in `https://bitbucket.org/gitia/zafra-pso-sarl`.

4 Testing Methodology and Results

In Section 2.1 the problem to be solved was presented, as a variant to GAP. A benchmarking suite is provided for this problem, and is available through `http://www.gitia.org/projects/vpgap/`. The benchmarking instances of this suite are identical to other standard suites for this kind of problems [Chu & Beasley, 1997, Yagiura et al., 1998], but with a smaller size (because the problem is much more difficult, as stated before in Section 2.1). The benchmarks were made with instances of types A, B, C and D, being D-type ones the hardest to solve, because c_{ij} and a_{ij} values are inversely proportional. In summary, 20 different instances were used, of low and medium complexity.

Benchmarks were executed on a single computer with a Intel Core i7-5500U @2.4 GHz microprocessor, 6 GB of RAM and a 64-bit Operating System. Ten independent executions of the PSO algorithm were made for each one of the instances, and the results are summarized in Table 1.

Table 1: PSO Results for the Proposed Problem.

Type	n	m	Opt	Particles	Iterations	Parameters	Found	t[s]	Confidence
A	3	12	0.658656	100	500	[0.8,1.5];1.43;1.43	0.658656	103.647	100%
A	4	10	0.497642	200	500	[0.8,1.5];1.43;1.43	0.497642	179.728	100%
A	5	10	0.544364	100	500	[0.8,1.5];1.43;1.43	0.544364	110.952	100%
A	3	15	0.871947	100	500	[0.8,1.5];1.43;1.43	0.871947	103.558	100%
A	4	12	0.628255	200	500	[0.8,1.5];1.43;1.43	0.628255	188.168	100%
B	3	12	0.543980	500	1000	[1.9,1.2];1.30;1.30	0.543980	908.177	100%
B	4	10	0.614256	100	500	[1.9,0.7];1.30;1.30	0.614256	99.546	90%
B	5	10	0.542515	600	1000	[1.9,1.5];1.30;1.30	0.542515	1145.751	80%
B	3	15	0.675879	100	1000	[1.9,0.7];1.30;1.30	0.675879	195.993	80%
B	4	12	0.606471	200	500	[1.9,0.7];1.30;1.30	0.606471	178.618	100%
C	3	12	0.564030	100	1000	[1.9,0.7];1.30;1.30	0.564030	157.779	100%
C	4	10	0.459722	200	1000	[1.9,0.7];1.30;1.30	0.459722	349.569	90%
C	5	10	0.473988	200	1500	[1.9,0.7];1.30;1.30	0.473988	499.459	90%
C	3	15	0.535528	300	1000	[1.9,0.7];1.30;1.30	0.535528	496.029	80%
C	4	12	0.501473	600	1000	[1.9,1.2];1.30;1.30	0.501473	1049.213	90%
D	3	12	0.277288	100	1000	[1.9,0.7];1.30;1.30	0.277288	197.566	100%
D	4	10	0.284239	1000	500	[1.9,1.2];1.30;1.30	0.284239	1318.644	90%
D	5	10	1.106979	300	1000	[1.9,0.7];1.30;1.30	1.106979	546.871	100%
D	3	15	0.308865	200	1000	[1.9,0.7];1.30;1.30	0.308865	356.877	80%
D	4	12	0.284524	300	2000	[1.2,1.9];1.30;1.30	0.284524	1052.803	50%

In Table 1, columns labeled *Type*, n and m indicate type of instance used, the number of agents and the number of tasks used, respectively. Column labeled *Opt* shows the optimal value as a reference, obtained previously for that particular instance in [Majorel Padilla et al., 2015]. The next three columns provide the number of particles used in the swarm, the number of iterations used by the simulations, and the PSO parameters used. In column labeled *Parameters*, three values are separated by semicolons: the range for inertial factor w, and the values for constants c_1 and c_2.

The best result obtained by the PSO algorithm for every instance is under the column labeled *Found*. It is remarkable that for all instances this value is exactly the same as the optimal value used as reference. The mean time required by the ten independent executions of the PSO algorithm to find the optimal value is also presented. Although these execution times may seem large enough for the simulation, the main objective of the tests was to prove that the algorithm can find the optimal solutions, without any timing restrictions. For example, the stopping criteria used was the number of iterations, for letting the particles to continue the searching of the solution's space even though they have reached the optimal value.

Finally, the last column of the table, labeled *Confidence*, shows how many times the PSO algorithm could find

the optimal value. This values should ideally be 100%, but as PSO algorithm can not guarantee its convergence, there is a probability that particles end up stuck at a local maximum. Nevertheless, for almost every instance this value is 80% or higher, with the exception of a D-Type instance with 4 agents and 12 tasks, for whom the optimal value could be found only in half of the tests.

5 Conclusions and Future Works

This paper introduced a new heuristic algorithm, a Particle Swarm Optimization implemented on a MultiAgent System, in order to solve a non-linear variant of the Generalized Assignment Problem. This variant was developed with the goal of optimizing sugar cane distribution in Tucumán, Argentina.

The algorithm was tested with 200 independent tests, part of a benchmark suite previously developed specially for this problem, and it could reach the optimal solution 91% of the times. We are comfortable to conclude that the algorithm can solve the problem, finding the optimal solution for all test instances (of different types). Moreover, results were significantly better than previous ones reported in [Majorel Padilla et al., 2015]. The execution times were larger than expected, but this is partially because we did not fully use the inherent parallel capabilities of MAS yet, and this is left as future work.

We wish to expand the results presented here in two different ways: first using larger test instances, for modeling real situations, and using the well known MAS scalability to run the simulations in a parallel environment.

Acknowledgments

This research was supported by Universidad Tecnológica Nacional under project UTI-UTN 1318.

References

[Blackwell, 2007] Blackwell, T. (2007). Particle swarm optimization in dynamic environments. In Yang, D. S., Ong, D. Y.-S., and Jin, D. Y., editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, number 51 in Studies in Computational Intelligence, pages 29–49. Springer Berlin Heidelberg.

[C. Rainwater & Romeijn, 2009] C. Rainwater, J. G. and Romeijn, E. H. (2009). The generalized assignment problem with flexible jobs. *Discrete Applied Mathematics*, 157(1):49–67.

[Cattrysse & Van Wassenhove, 1992] Cattrysse, D. G. and Van Wassenhove, L. N. (1992). A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60(3):260–272.

[Chu & Beasley, 1997] Chu, P. C. and Beasley, J. E. (1997). A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, 24(1):17–23.

[Clerc & Kennedy, 2002] Clerc, M. & Kennedy, J. (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73.

[Fisher et al., 1986] Fisher, M. L., Jaikumar, R., and Van Wassenhove, L. N. (1986). A multiplier adjustment method for the generalized assignment problem. *Management Science*, 32(9):1095–1103.

[Giles et al., 2009] Giles, R., Lyne, P., Venter, R., Niekerk, J., Dines, G., et al. (2009). Vehicle scheduling project success at south african and swaziland sugar mills. In *Proceedings of the Annual Congress-South African Sugar Technologists' Association*, pages 151–163. South African Sugar Technologists' Association.

[Hu et al., 2004] Hu, X., Shi, Y., and Eberhart, R. (2004). Recent advances in particle swarm. In *Congress on Evolutionary Computation, 2004. CEC2004*, volume 1, pages 90–97 Vol.1.

[Kennedy & Eberhart, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *, IEEE International Conference on Neural Networks, 1995. Proceedings*, volume 4, pages 1942–1948 vol.4.

[Kennedy & Eberhart, 1997] Kennedy, J. and Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *, 1997 IEEE International Conference on Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation*, volume 5, pages 4104–4108 vol.5.

[Krumke & Thielen, 2013] Krumke, S. O. and Thielen, C. (2013). The generalized assignment problem with minimum quantities. *European Journal of Operational Research*, 228(1):46–55.

[Kumar et al., 2011] Kumar, R., Sharma, D., and Sadu, A. (2011). A hybrid multi-agent based particle swarm optimization algorithm for economic power dispatch. *International Journal of Electrical Power & Energy Systems*, 33(1):115–123.

[Laguna et al., 1995] Laguna, M., Kelly, J. P., Gonzlez-Velarde, J., and Glover, F. (1995). Tabu search for the multilevel generalized assignment problem. *European Journal of Operational Research*, 82(1):176–189.

[Le Gal et al., 2009] Le Gal, P. Y., Le Masson, J., Bezuidenhout, C. N., and Lagrange, L. F. (2009). Coupled modelling of sugarcane supply planning and logistics as a management tool. *Computers and Electronics in Agriculture*, 68(2):168–177.

[Majorel Padilla et al., 2015] Majorel Padilla, N., Araujo, P., Will, A., and Rodriguez, S. (2015). Modelizacin no lineal del problema de la distribucin de la caa de azcar y su implementacin en sistemas multiagentes organizacionales. In *3er Congreso Nacional de Ingeniera Informatica/Sistemas de Informatica*, Buenos Aires, Argentina.

[Martello & Toth, 1990] Martello, S. and Toth, P. (1990). *Knapsack problems*. Wiley New York.

[Osman, 1995] Osman, I. H. (1995). Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. *Operations-Research-Spektrum*, 17(4):211–225.

[Ozbakir et al., 2010] Ozbakir, L., Baykasolu, A., and Tapkan, P. (2010). Bees algorithm for generalized assignment problem. *Applied Mathematics and Computation*, 215(11):3782–3795.

[Pugh & Martinoli, 2007] Pugh, J. and Martinoli, A. (2007). Inspiring and modeling multi-robot search with particle swarm optimization. In *2007 IEEE Swarm Intelligence Symposium*, pages 332–339.

[Ramalhinho & Serra, 2008] Ramalhinho, H. and Serra, D. (2008). Adaptive search heuristics for the generalized assignment problem. *Mathware & soft computing*, 9(3):209–234.

[Rodriguez et al., 2014] Rodriguez, S., Gaud, N., and Galland, S. (2014). Sarl: A general-purpose agent-oriented programming language. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 3, pages 103–110.

[Shi & Eberhart, 1998] Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. In *, The 1998 IEEE International Conference on Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence*, pages 69–73.

[Shi & Eberhart, 1999] Shi, Y. and Eberhart, R. C. (1999). Empirical study of particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99*, volume 3, page 1950 Vol. 3.

[Van Den Bergh, 2002] Van Den Bergh, F. (2002). *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, Pretoria, South Africa.

[Wang et al., 2010] Wang, Z., Yang, R., and Wang, L. (2010). Multi-agent control system with intelligent optimization for smart and energy-efficient buildings. In *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, pages 1144–1149.

[Weiss, 2013] Weiss, G. (2013). *Multiagent Systems*. The MIT Press, Cambridge, Massachusetts, second edition.

[Wooldridge, 2009] Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. Wiley, Chichester, U.K, 2nd edition.

[Yagiura et al., 1998] Yagiura, M., Yamaguchi, T., and Ibaraki, T. (1998). A variable depth search algorithm with branching search for the generalized assignment problem. *Optimization Methods and Software*, 10(2):419–441.

[Yosnual & Supsomboon, 2004] Yosnual, J. and Supsomboon, S. (2004). An integer programming for sugarcane factory supply allocation. In *Proceedings of the fifth Asia pacific industrial engineering and management systems conference*, volume 21, pages 1–21.

[Zhang et al., 2003] Zhang, L. B., Zhou, C. G., Liu, X. H., Ma, Z. Q., Ma, M., and Liang, Y. C. (2003). Solving multi objective optimization problems using particle swarm optimization. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03*, volume 4, pages 2400–2405 Vol.4.

[Zhao et al., 2005] Zhao, B., Guo, C. X., and Cao, Y. J. (2005). A multiagent-based particle swarm optimization approach for optimal reactive power dispatch. *IEEE Transactions on Power Systems*, 20(2):1070–1078.